

ACL-08: HLT

**46th  
Annual Meeting  
of the Association for  
Computational Linguistics:  
Human Language  
Technologies**

**Proceedings of the Conference**

June 15–20, 2008  
The Ohio State University  
Columbus, Ohio, USA

Production and Manufacturing by  
*Omnipress Inc.*  
2600 Anderson Street  
Madison, WI 53707  
USA

Microsoft  
**Research**



**BBN**  
TECHNOLOGIES



**Powerset**



©2008 The Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-932432-04-6

## Table of Contents

Preface: General Chair .....	xi
Preface: Program Chairs .....	xii
Organizers .....	xv
Program Committee .....	xvii
Conference Program .....	xxi
<i>Mining Wiki Resources for Multilingual Named Entity Recognition</i> Alexander E. Richman and Patrick Schone .....	1
<i>Distributional Identification of Non-Referential Pronouns</i> Shane Bergsma, Dekang Lin and Randy Goebel .....	10
<i>Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs</i> Marius Paşca and Benjamin Van Durme .....	19
<i>The Tradeoffs Between Open and Traditional Relation Extraction</i> Michele Banko and Oren Etzioni .....	28
<i>PDT 2.0 Requirements on a Query Language</i> Jiří Mírovský .....	37
<i>Task-oriented Evaluation of Syntactic Parsers and Their Representations</i> Yusuke Miyao, Rune Sætne, Kenji Sagae, Takuya Matsuzaki and Jun'ichi Tsujii .....	46
<i>MAXSIM: A Maximum Similarity Metric for Machine Translation Evaluation</i> Yee Seng Chan and Hwee Tou Ng .....	55
<i>Contradictions and Justifications: Extensions to the Textual Entailment Task</i> Ellen M. Voorhees .....	63
<i>Cohesive Phrase-Based Decoding for Statistical Machine Translation</i> Colin Cherry .....	72
<i>Phrase Table Training for Precision and Recall: What Makes a Good Phrase and a Good Phrase Pair?</i> Yonggang Deng, Jia Xu and Yuqing Gao .....	81
<i>Measure Word Generation for English-Chinese SMT Systems</i> Dongdong Zhang, Mu Li, Nan Duan, Chi-Ho Li and Ming Zhou .....	89
<i>Bayesian Learning of Non-Compositional Phrases with Synchronous Parsing</i> Hao Zhang, Chris Quirk, Robert C. Moore and Daniel Gildea .....	97

<i>Applying a Grammar-Based Language Model to a Simplified Broadcast-News Transcription Task</i> Tobias Kaufmann and Beat Pfister .....	106
<i>Automatic Editing in a Back-End Speech-to-Text System</i> Maximilian Bisani, Paul Vozila, Olivier Divay and Jeff Adams .....	114
<i>Grounded Language Modeling for Automatic Speech Recognition of Sports Video</i> Michael Fleischman and Deb Roy .....	121
<i>Lexicalized Phonotactic Word Segmentation</i> Margaret M. Fleck .....	130
<i>A Re-examination of Query Expansion Using Lexical Resources</i> Hui Fang .....	139
<i>Selecting Query Term Alternations for Web Search by Exploiting Query Contexts</i> Guihong Cao, Stephen Robertson and Jian-Yun Nie .....	148
<i>Searching Questions by Identifying Question Topic and Question Focus</i> Huizhong Duan, Yunbo Cao, Chin-Yew Lin and Yong Yu .....	156
<i>Trainable Generation of Big-Five Personality Styles through Data-Driven Parameter Estimation</i> François Mairesse and Marilyn Walker .....	165
<i>Correcting Misuse of Verb Forms</i> John Lee and Stephanie Seneff .....	174
<i>Hypertagging: Supertagging for Surface Realization with CCG</i> Dominic Espinosa, Michael White and Dennis Mehay .....	183
<i>Forest-Based Translation</i> Haitao Mi, Liang Huang and Qun Liu .....	192
<i>A Discriminative Latent Variable Model for Statistical Machine Translation</i> Phil Blunsom, Trevor Cohn and Miles Osborne .....	200
<i>Efficient Multi-Pass Decoding for Synchronous Context Free Grammars</i> Hao Zhang and Daniel Gildea .....	209
<i>Regular Tree Grammars as a Formalism for Scope Underspecification</i> Alexander Koller, Michaela Regneri and Stefan Thater .....	218
<i>Classification of Semantic Relationships between Nominals Using Pattern Clusters</i> Dmitry Davidov and Ari Rappoport .....	227
<i>Vector-based Models of Semantic Composition</i> Jeff Mitchell and Mirella Lapata .....	236
<i>Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition</i> Andrew Arnold, Ramesh Nallapati and William W. Cohen .....	245

<i>Refining Event Extraction through Cross-Document Inference</i> Heng Ji and Ralph Grishman .....	254
<i>Learning Document-Level Semantic Properties from Free-Text Annotations</i> S.R.K. Branavan, Harr Chen, Jacob Eisenstein and Regina Barzilay .....	263
<i>Automatic Image Annotation Using Auxiliary Text Information</i> Yansong Feng and Mirella Lapata .....	272
<i>Hedge Classification in Biomedical Texts with a Weakly Supervised Selection of Keywords</i> György Szarvas .....	281
<i>When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging</i> Alina Andreevskaia and Sabine Bergler .....	290
<i>A Generic Sentence Trimmer with CRFs</i> Tadashi Nomoto .....	299
<i>A Joint Model of Text and Aspect Ratings for Sentiment Summarization</i> Ivan Titov and Ryan McDonald .....	308
<i>Improving Parsing and PP Attachment Performance with Sense Information</i> Eneko Agirre, Timothy Baldwin and David Martinez .....	317
<i>A Logical Basis for the D Combinator and Normal Form in CCG</i> Frederick Hoyt and Jason Baldridge .....	326
<i>Parsing Noun Phrase Structure with CCG</i> David Vadas and James R. Curran .....	335
<i>Sentence Simplification for Semantic Role Labeling</i> David Vickrey and Daphne Koller .....	344
<i>Summarizing Emails with Conversational Cohesion and Subjectivity</i> Giuseppe Carenini, Raymond T. Ng and Xiaodong Zhou .....	353
<i>Ad Hoc Treebank Structures</i> Markus Dickinson .....	362
<i>A Single Generative Model for Joint Morphological Segmentation and Syntactic Parsing</i> Yoav Goldberg and Reut Tsarfaty .....	371
<i>Which Words Are Hard to Recognize? Prosodic, Lexical, and Disfluency Factors that Increase ASR Error Rates</i> Sharon Goldwater, Dan Jurafsky and Christopher D. Manning .....	380
<i>Name Translation in Statistical Machine Translation - Learning When to Transliterate</i> Ulf Hermjakob, Kevin Knight and Hal Daumé III .....	389

<i>Using Adaptor Grammars to Identify Synergies in the Unsupervised Acquisition of Linguistic Structure</i> Mark Johnson .....	398
<i>Inducing Gazetteers for Named Entity Recognition by Large-Scale Clustering of Dependency Relations</i> Jun'ichi Kazama and Kentaro Torisawa .....	407
<i>Evaluating Roget's Thesauri</i> Alistair Kennedy and Stan Szpakowicz .....	416
<i>Unsupervised Translation Induction for Chinese Abbreviations using Monolingual Corpora</i> Zhifei Li and David Yarowsky .....	425
<i>Which Are the Best Features for Automatic Verb Classification</i> Jianguo Li and Chris Brew .....	434
<i>Collecting a Why-Question Corpus for Development and Evaluation of an Automatic QA-System</i> Joanna Mrozinski, Edward Whittaker and Sadaoki Furui .....	443
<i>Solving Relational Similarity Problems Using the Web as a Corpus</i> Preslav Nakov and Marti A. Hearst .....	452
<i>Combining Speech Retrieval Results with Generalized Additive Models</i> J. Scott Olsson and Douglas W. Oard .....	461
<i>A Critical Reassessment of Evaluation Baselines for Speech Summarization</i> Gerald Penn and Xiaodan Zhu .....	470
<i>Intensional Summaries as Cooperative Responses in Dialogue: Automation and Evaluation</i> Joseph Polifroni and Marilyn Walker .....	479
<i>Word Clustering and Word Selection Based Feature Reduction for MaxEnt Based Hindi NER</i> Sujan Kumar Saha, Pabitra Mitra and Sudeshna Sarkar .....	488
<i>Combining EM Training and the MDL Principle for an Automatic Verb Classification Incorporating Selectional Preferences</i> Sabine Schulte im Walde, Christian Hying, Christian Scheible and Helmut Schmid .....	496
<i>Randomized Language Models via Perfect Hash Functions</i> David Talbot and Thorsten Brants .....	505
<i>Applying Morphology Generation Models to Machine Translation</i> Kristina Toutanova, Hisami Suzuki and Achim Ruopp .....	514
<i>Multilingual Harvesting of Cross-Cultural Stereotypes</i> Tony Veale, Yanfen Hao and Guofu Li .....	523
<i>Semi-Supervised Convex Training for Dependency Parsing</i> Qin Iris Wang, Dale Schuurmans and Dekang Lin .....	532

<i>Chinese-English Backward Transliteration Assisted with Mining Monolingual Web Pages</i> Fan Yang, Jun Zhao, Bo Zou, Kang Liu and Feifan Liu .....	541
<i>Robustness and Generalization of Role Sets: PropBank vs. VerbNet</i> Beñat Zepirain, Eneko Agirre and Lluís Màrquez .....	550
<i>A Tree Sequence Alignment-based Tree-to-Tree Translation Model</i> Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan and Sheng Li .....	559
<i>Automatic Syllabification with Structured SVMs for Letter-to-Phoneme Conversion</i> Susan Bartlett, Grzegorz Kondrak and Colin Cherry .....	568
<i>A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model</i> Libin Shen, Jinxi Xu and Ralph Weischedel .....	577
<i>Forest Reranking: Discriminative Parsing with Non-Local Features</i> Liang Huang .....	586
<i>Simple Semi-supervised Dependency Parsing</i> Terry Koo, Xavier Carreras and Michael Collins .....	595
<i>Optimal k-arization of Synchronous Tree-Adjoining Grammar</i> Rebecca Nesson, Giorgio Satta and Stuart M. Shieber .....	604
<i>Enhancing Performance of Lexicalised Grammars</i> Rebecca Dridan, Valia Kordoni and Jeremy Nicholson .....	613
<i>Assessing Dialog System User Simulation Evaluation Measures Using Human Judges</i> Hua Ai and Diane J. Litman .....	622
<i>Robust Dialog Management with N-Best Hypotheses Using Dialog Examples and Agenda</i> Cheongjae Lee, Sangkeun Jung and Gary Geunbae Lee .....	630
<i>Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz Data: Bootstrapping and Evaluation</i> Verena Rieser and Oliver Lemon .....	638
<i>Phrase Chunking Using Entropy Guided Transformation Learning</i> Ruy Luiz Milidiú, Cícero Nogueira dos Santos and Julio C. Duarte .....	647
<i>Learning Bigrams from Unigrams</i> Xiaojin Zhu, Andrew B. Goldberg, Michael Rabbat and Robert Nowak .....	656
<i>Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data</i> Jun Suzuki and Hideki Isozaki .....	665
<i>Large Scale Acquisition of Paraphrases for Learning Surface Patterns</i> Rahul Bhagat and Deepak Ravichandran .....	674

<i>Contextual Preferences</i>	
Idan Szpektor, Ido Dagan, Roy Bar-Haim and Jacob Goldberger .....	683
<i>Unsupervised Discovery of Generic Relationships Using Pattern Clusters and its Evaluation by Automatically Generated SAT Analogy Questions</i>	
Dmitry Davidov and Ari Rappoport .....	692
<i>Improving Search Results Quality by Customizing Summary Lengths</i>	
Michael Kaisser, Marti A. Hearst and John B. Lowe .....	701
<i>Using Conditional Random Fields to Extract Contexts and Answers of Questions from Online Forums</i>	
Shilin Ding, Gao Cong, Chin-Yew Lin and Xiaoyan Zhu .....	710
<i>Learning to Rank Answers on Large Online QA Collections</i>	
Mihai Surdeanu, Massimiliano Ciaramita and Hugo Zaragoza .....	719
<i>Unsupervised Lexicon-Based Resolution of Unknown Words for Full Morphological Analysis</i>	
Meni Adler, Yoav Goldberg, David Gabay and Michael Elhadad .....	728
<i>Unsupervised Multilingual Learning for Morphological Segmentation</i>	
Benjamin Snyder and Regina Barzilay .....	737
<i>EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start)</i>	
Yoav Goldberg, Meni Adler and Michael Elhadad .....	746
<i>Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation</i>	
Jakob Uszkoreit and Thorsten Brants .....	755
<i>Enriching Morphologically Poor Languages for Statistical Machine Translation</i>	
Eleftherios Avramidis and Philipp Koehn .....	763
<i>Learning Bilingual Lexicons from Monolingual Corpora</i>	
Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick and Dan Klein .....	771
<i>Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora</i>	
Shiqi Zhao, Haifeng Wang, Ting Liu and Sheng Li .....	780
<i>Unsupervised Learning of Narrative Event Chains</i>	
Nathanael Chambers and Dan Jurafsky .....	789
<i>Semantic Role Labeling Systems for Arabic using Kernel Methods</i>	
Mona Diab, Alessandro Moschitti and Daniele Pighin .....	798
<i>An Unsupervised Approach to Biography Production Using Wikipedia</i>	
Fadi Biadisy, Julia Hirschberg and Elena Filatova .....	807
<i>Generating Impact-Based Summaries for Scientific Literature</i>	
Qiaozhu Mei and ChengXiang Zhai .....	816



<i>Can You Summarize This? Identifying Correlates of Input Difficulty for Multi-Document Summarization</i> Ani Nenkova and Annie Louis .....	825
<i>You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement</i> Micha Elsner and Eugene Charniak .....	834
<i>An Entity-Mention Model for Coreference Resolution with Inductive Logic Programming</i> Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, Ting Liu and Sheng Li .....	843
<i>Gestural Cohesion for Topic Segmentation</i> Jacob Eisenstein, Regina Barzilay and Randall Davis .....	852
<i>Multi-Task Active Learning for Linguistic Annotations</i> Roi Reichart, Katrin Tomanek, Udo Hahn and Ari Rappoport .....	861
<i>Generalized Expectation Criteria for Semi-Supervised Learning of Conditional Random Fields</i> Gideon S. Mann and Andrew McCallum .....	870
<i>Analyzing the Errors of Unsupervised Learning</i> Percy Liang and Dan Klein .....	879
<i>Joint Word Segmentation and POS Tagging Using a Single Perceptron</i> Yue Zhang and Stephen Clark .....	888
<i>A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging</i> Wenbin Jiang, Liang Huang, Qun Liu and Yajuan Lü .....	897
<i>Joint Processing and Discriminative Training for Letter-to-Phoneme Conversion</i> Sittichai Jiampojarn, Colin Cherry and Grzegorz Kondrak .....	905
<i>A Probabilistic Model for Fine-Grained Expert Search</i> Shenghua Bao, Huizhong Duan, Qi Zhou, Miao Xiong, Yunbo Cao and Yong Yu .....	914
<i>Credibility Improves Topical Blog Post Retrieval</i> Wouter Weerkamp and Maarten de Rijke .....	923
<i>Linguistically Motivated Features for Enhanced Back-of-the-Book Indexing</i> Andras Csomai and Rada Mihalcea .....	932
<i>Resolving Personal Names in Email Using Context Expansion</i> Tamer Elsayed, Douglas W. Oard and Galileo Namata .....	941
<i>Integrating Graph-Based and Transition-Based Dependency Parsers</i> Joakim Nivre and Ryan McDonald .....	950
<i>Efficient, Feature-based, Conditional Random Field Parsing</i> Jenny Rose Finkel, Alex Kleeman and Christopher D. Manning .....	959
<i>A Deductive Approach to Dependency Parsing</i> Carlos Gómez-Rodríguez, John Carroll and David Weir .....	968

<i>Evaluating a Crosslinguistic Grammar Resource: A Case Study of Wambaya</i> Emily M. Bender .....	977
<i>Better Alignments = Better Translations?</i> Kuzman Ganchev, João V. Graça and Ben Taskar .....	986
<i>Mining Parenthetical Translations from the Web by Word Alignment</i> Dekang Lin, Shaojun Zhao, Benjamin Van Durme and Marius Paşca .....	994
<i>Soft Syntactic Constraints for Hierarchical Phrased-Based Translation</i> Yuval Marton and Philip Resnik .....	1003
<i>Generalizing Word Lattice Translation</i> Christopher Dyer, Smaranda Muresan and Philip Resnik .....	1012
<i>Combining Multiple Resources to Improve SMT-based Paraphrasing Model</i> Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu and Sheng Li .....	1021
<i>Extraction of Entailed Semantic Relations Through Syntax-Based Comma Resolution</i> Vivek Srikumar, Roi Reichart, Mark Sammons, Ari Rappoport and Dan Roth .....	1030
<i>Finding Contradictions in Text</i> Marie-Catherine de Marneffe, Anna N. Rafferty and Christopher D. Manning .....	1039
<i>Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs</i> Zornitsa Kozareva, Ellen Riloff and Eduard Hovy .....	1048

## Preface: General Chair

I am honored to be serving as General Conference Chair for the annual conference in our field. This year's conference, ACL-08: HLT, is jointly sponsored by the Association for Computational Linguistics and the North American Chapter of the Association for Computational Linguistics and it thus brings together the traditions of both organizations. As is evident from the title, one of those traditions is the focus on research from all areas of Human Language Technology, including information retrieval, natural language processing and speech. The conference features invited speakers in speech and information retrieval and there are sessions devoted to all three of these areas. I hope this conference will again encourage interaction among researchers from the different areas.

Since I was last involved in organizing the ACL Conferences back in the 90's, the conferences have grown dramatically. I was surprised to learn the number of people required to make the conference happen. Some 30 odd people are serving in Chair or Co-Chair capacity of various aspects of the conference. While I was pleased to have the opportunity of shaping aspects of the conference, I have to say that the real bulk of the work is done by the many Chairs involved. So I want to express my gratitude to all of them for their commitment and dedication to making sure that all ran smoothly. I am impressed by the energy and time that everyone gave to this volunteer activity.

I would like to thank the Program Chairs, Johanna Moore, Simone Teufel, James Allan and Sadaoki Furui, who have put in many hours to provide us with the main program for the conference and the Local Arrangements Chair, Chris Brew, who has provided us with the venue for the conference and oversaw the many time-demanding details. DJ Hovermale also put in many hours as webmaster, collecting information from everyone. I would like to thank the Chairs of the Student Research Workshop, Ebru Arisoy, Wolfgang Maier and Keisuke Inoue, who worked quite independently, along with the Faculty Advisor, Jan Wiebe. The Workshop Chair, Ming Zhou, managed the workshop program with ease, a program that has grown over the years so that it seems like a conference in and of itself. The Tutorial Chairs, Ani Nenkova, Marilyn Walker and Eugene Agichtein, have put together a fine tutorial program and the Demo Chair, Jimmy Lin, has organized a nice series of demos. The Sponsorship Chairs are responsible for bringing in funding to cover various programs and I would like to thank Inderjeet Mani, Josef van Genabith and Michael White for their efforts in this regard. The Publicity Chairs, Hal Daumé III, Eric Fosler-Lussier and Diane Kelly, reached out to communities outside the central natural language areas to encourage people to submit papers and attend the conference. Finally, I would like to give a big thanks to the Publication Chairs, Joakim Nivre and Noah A. Smith, who were very organized and handily managed the job of pulling all materials together for the main conference and workshop proceedings, no small feat.

In addition to the Chairs, individuals within the ACL organization itself deserve recognition. First and foremost, my thanks goes to Dragomir Radev, who provided guidance about what to do next at every step and who had the answer to every question I had within seconds. Owen Rambow also provided much needed advice from the perspective of the North American Chapter. Priscilla Rasmussen is critical to the running of the conference, with her organizational history of how things work. Finally, I would like to thank the Coordinating Committee for being available for discussion and for providing advice.

Kathleen McKeown  
ACL-08: HLT General Chair

## Preface: Program Chairs

The program for ACL-08: HLT features a wide variety of avenues for authors to present their latest work in computational linguistics, information retrieval, and speech technology. The program includes: full papers, short papers, posters, demonstrations, and a student research workshop, as well as pre- and post-conference tutorials and workshops. In our program design, we attempted to combine the successful approach of ACL07, which had four parallel oral sessions of 25-min full paper presentations, with the HLT model of presenting late-breaking results in parallel sessions of 15-min short paper presentations. We also experimented with an idea adopted from Interspeech, in which authors can choose their desired mode of presentation, oral or poster, based on their assessment of how best to present their work. There is no distinction between posters and oral presentations in terms of quality or in terms of how they appear in the Proceedings. Although it will take more than one year to see this change fully taken up by the membership, we were happy to see some authors choose the poster option from the very outset. Area chairs also used their discretion in indicating which submissions would benefit from which mode of presentation. If the number of submissions continues to grow as it has done in the past few years, poster sessions will be one way to managing this growth without creating a large number of parallel sessions.

This year, the program committee received yet another record-breaking number of submissions, with 470 full and 275 short paper submissions. Full papers were due in mid-January, and the program committee accepted 119 (25%) of these, 95 as oral presentations and 24 as posters. Short papers were due in mid-March, and the committee accepted 64 (23%) of these, 32 for oral presentation and 32 for poster presentation.

First and foremost, we thank all the authors for submitting papers describing their recent work; the sheer number of submissions reflects how active our field is. We are greatly indebted to the 34 area chairs who recruited 720 reviewers, and who managed the reviewing process of both full and short papers in their areas. Reviewers wrote three reviews for each full paper submission, and two reviews for each short paper submission, for a staggering total of just under 2000 reviews! Miraculously, there were only a handful of late reviews. Well done everyone!

As the number of submissions and, consequently the number of area chairs, has risen over the last few years, the ACL program committee has moved away from having a face-to-face meeting of all area chairs. For ACL08: HLT, two of the program co-chairs met for two days at Edinburgh University, using email and teleconferencing to get input from the two program co-chairs not based in Europe, and all of the area chairs. For short paper decision making, three of the four program co-chairs held a teleconference, with input from the fourth co-chair by email as time zone differences permitted.

Another first this year was our decision to award several outstanding paper prizes, rather than trying to identify a single best paper. We did this because we felt that it is typical for conferences as large as this to have several particularly exciting, innovative, and well-crafted papers, and it is extremely difficult to compare quality across areas. We asked area chairs to nominate papers for the various awards and then formed an Outstanding Paper Committee, who wish to remain anonymous, and to whom we owe a great debt of gratitude for their hard work at short notice.

As usual, the main program will run for three days: there will be four parallel sessions of paper presentations. One of these is devoted to the Student Research Workshop, which we would like to thank Ebru Abrisoy, Wolfgang Maier and Keisuke Inoue for organizing. There will also be a poster session on Monday evening, with food and drink to keep everyone going. The demo session, organized by Jimmy Lin, will be held concurrently with the poster session. This year there will be five plenary sessions: two for our very distinguished invited speakers, Susan Dumais and Marc Swerts, one for presentation of the four outstanding papers, one for the presentation by this year's Lifetime Achievement Award winner, and finally one for the ACL business meeting.

Also as usual the conference is flanked by tutorial sessions and workshops. We would like to thank Ani Nenkova, Marilyn Walker and Eugene Agichtein for organizing the tutorials, and Ming Zhou, ChengXiang Zhai and Helen Meng for compiling an excellent program of workshops.

We also thank Kathy McKeown, General Conference Chair, the Local Arrangements Committee headed by Chris Brew, the ACL executive committee, for their help and advice, and last year's co-chairs, Antal van den Bosch and Annie Zaenen, for sharing their experience.

Finally, there were three things that made this all possible. First, we were helped immensely by Jason Eisner, who has compiled an excellent web site on "How to Serve as Program Chair of a Conference" (<http://www.cs.jhu.edu/~jason/advice/how-to-chair-a-conference.html>). This saved us more than once! Second, we employed a recent PhD, James Clarke, to help us get started with START, and to simply deal with the large volume of work that must be processed within the first few days after submissions are received. James kept us sane. Third, there is the invaluable START system for managing paper submission, reviewing, and decision making. We owe Rich Gerber and the START team a million thanks for responding to questions quickly, and even modifying START overnight to provide what we asked for.

Our most sincere thanks go to Joakim Nivre and Noah A. Smith who took all of our labors and put together the wonderful Proceedings you are now reading.

We hope you enjoy the conference,

Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui  
ACL-08: HLT Program Chairs



# Organizers

## General Chair:

Kathleen McKeown, Columbia University, USA

## Local Arrangements Chair:

Chris Brew, The Ohio State University, USA

## Program Chairs:

Johanna Moore (Natural Language Processing), University of Edinburgh, UK

Simone Teufel (Natural Language Processing), University of Cambridge, UK

James Allan (Information Retrieval), University of Massachusetts, USA

Sadaoki Furui (Speech), Tokyo Institute of Technology, Japan

## Student Research Workshop:

Ebru Arisoy (Speech co-chair), Bogazici University, Turkey

Wolfgang Maier (Natural Language Processing co-chair), University of Tuebingen, Germany

Keisuke Inoue (Information Retrieval co-chair), Syracuse University, USA

Janyce Wiebe (Faculty Advisor), University of Pittsburgh, USA

## Workshop Chair:

Ming Zhou, Microsoft Research China, China

## Tutorial Chairs:

Ani Nenkova (Coordinator), University of Pennsylvania, USA

Marilyn Walker, University of Sheffield, UK

Eugene Agichtein, Emory University, USA

## Demo Chair:

Jimmy Lin, University of Maryland, USA

## Sponsorship Chairs:

Inderjeet Mani, Mitre Corporation, USA

Josef van Genabith, Dublin City University, Ireland

Michael White, The Ohio State University, USA

**Publications Chairs:**

Joakim Nivre, Växjö University and Uppsala University, Sweden  
Noah Smith, Carnegie Mellon University, USA

**Publicity Chairs:**

Hal Daumé III, University of Utah, USA  
Eric Fosler-Lussier, The Ohio State University, USA  
Diane Kelly, University of North Carolina, USA

**Student Volunteers:**

Ilana Bromberg (Volunteer co-ordinator)  
Crystal Nakatsu (Accommodation requests)  
Dominic Espinosa (Conference booklet)

**Webmaster:**

DJ Hovermale, The Ohio State University, USA

**Publications Committee:**

Marco Kuhlmann, Uppsala University, Sweden  
Carol Sisson, Carnegie Mellon University, USA  
Filip Salomonsson, Uppsala University, Sweden

**Registration:**

Priscilla Rasmussen, Association for Computational Linguistics (ACL)

**ACL Coordinating Committee:**

Nicoletta Calzolari, Università di Pisa, Italy  
Jennifer Chu-Carroll, IBM, USA  
Graeme Hirst, University of Toronto, Canada  
Chris Manning, Stanford University, USA  
Kathleen McCoy, University of Delaware, US  
Dragomir Radev, University of Michigan, USA  
Owen Rambow, Columbia University, USA  
Priscilla Rasmussen, Association for Computational Linguistics (ACL)  
Mark Steedman, The University of Edinburgh, UK  
Suzanne Stevenson, University of Toronto, Canada



# Program Committee

## Program Chairs:

Johanna D. Moore, University of Edinburgh (UK)  
Simone Teufel, Cambridge University (UK)  
James Allan, University of Massachusetts Amherst (USA)  
Sadaoki Furui, Tokyo Institute of Technology (Japan)

## Area Chairs:

Jason Baldridge, University of Texas at Austin (USA)  
Regina Barzilay, Massachusetts Institute of Technology (USA)  
Pushpak Bhattacharaya, Indian Institute of Technology Bombay (India)  
David Carmel, IBM Research (Israel)  
David Chiang, USC/Information Sciences Institute (USA)  
Steve Clark, Oxford University (UK)  
Hal Daumé III, University of Utah (USA)  
Dina Demner-Fushman, National Library of Medicine (USA)  
Li Deng, Microsoft Research (USA)  
Mark Dras, Macquarie University (Australia)  
Pascale Fung, Hong Kong University of Science and Technology (China)  
Daniel Gildea, University of Rochester (USA)  
John Hansen, University of Texas at Dallas (USA)  
Daniel Hardt, Copenhagen Business School (Denmark)  
Masato Ishizaki, University of Tokyo (Japan)  
Michael Johnston, AT&T Labs Reserach (USA)  
Min-Yen Kan, National University of Singapore (Singapore)  
Noriko Kando, National Institute of Informatics (Japan)  
Emiel Krahmer, Tilburg University (Netherlands)  
Elizabeth Liddy, Syracuse University (USA)  
Chin-Yew Lin, Microsoft Research Asia (China)  
Andrew McCallum, University of Massachusetts Amherst (USA)  
Katja Markert, University of Leeds (UK)  
Lluís Màrquez, Universitat Politècnica de Catalunya (Spain)  
Raymond Mooney, University of Texas at Austin (USA)  
Rashmi Prasad, University of Pennsylvania (USA)  
Helmut Schmid, University of Stuttgart (Germany)  
Sabine Schulte im Walde, University of Stuttgart (Germany)  
Rohini Srihari, University of Buffalo (USA)  
Manfred Stede, Potsdam University (Germany)  
Keiichi Tokuda, Nagoya Institute of Technology (Japan)  
Taro Watanabe, NTT Communication Science Laboratories (Japan)  
Janyce Wiebe, University of Pittsburgh (USA)  
David Weir, Sussex University (UK)

## **Program Committee Members:**

Doug Appelt, Steven Abney, Meni Adler, Stergos Afantenos, Eugene Agichtein, Eneko Agirre, Lars Ahrenberg, Salah Ait-Mokhtar, Ahmet Aker, Jan Alexandersson, Afra Alishahi, Yasemin Altun, Sophia Ananiadou, Galen Andrew, Masahiro Araki, Masayuki Asahara, Nicholas Asher, Michaela Atterer, Necip Fazil Ayan

Timothy Baldwin, Srinivas Bangalore, Michele Banko, Colin Bannard, Roy Bar-Haim, Marco Baroni, Roberto Basili, John Bateman, Johnathan Baxter, Tilman Becker, Ron Bekkerman, Anja Belz, José Benedí, Paul Bennett, Sabine Bergler, Kay Berklings, Yves Bestgen, Rahul Bhagat, Indrajit Bhattacharya, Tanmay Bhattacharya, Pushpak Bhattacharyya, Chris Biemann, Dan Bikel, Mikhail Bilenko, Jeff Bilmes, Philippe Blache, Patrick Blackburn, Sasha Blair-Goldensohn, David Blei, John Blitzer, Phil Blunsom, Gemma Boleda, Johan Bos, Pierre Boullier, Karl Branting, Thorsten Brants, Eric Breck, Chris Brew, Ted Briscoe, Chris Brockett, Ralf Brown, Paul Buitelaar, Razvan Bunescu, Harry Bunt, Stephan Busemann, Donna Byron

Aoife Cahill, Charles Callaway, Chris Callison-Burch, Nicoletta Calzolari, Nick Campbell, Yunbo Cao, Sandra Carberry, Giuseppe Carenini, Jean Carletta, David Carmel, Xavier Carreras, John Carroll, Francisco Casacuberta, Justine Cassell, Lawrence Cavedon, Suleyman Cetintas, Yee Seng Chan, Raman Chandrasekar, Jason Chang, Eugene Charniak, Wanxiang Che, Ciprian Chelba, Hsin-Hsi Chen, John Chen, Colin Cherry, David Chiang, Christian Chiarcos, Yejin Choi, Min Chu, Tat-Seng Chua, Jennifer Chu-Carroll, Ken Church, Massimiliano Ciaramita, Philip Cimiano, Ariel Cohen, Trevor Cohn, Michael Collins, Alistair Conkie, John Conroy, Robin Cooper, Bonaventura Coppola, Mark Core, Marta Costa-jussá, Koby Crammer, Mark Craven, Josep Crego, Silviu Cucerzan, Hang Cui, Aron Culotta, James Curran

Walter Daelemans, Ido Dagan, Robert Dale, Hoa Dang, Hal Daumé III, Eric de la Clergerie, Maarten de Rijke, Vera Demberg, Dina Demner-Fushman, Yasuharu Den, Steve DeNeefe, John DeNero, Li Deng, Yonggang Deng, Ann Devitt, Barbara di Eugenio, Mona Diab, Fernando Diaz, Anne Diekema, Giuseppe DiFabrizio, Kohji Dohsaka, Bill Dolan, Bonnie Dorr, John Dowding, Mark Dras, Mark Dredze, Gregory Druck, Amit Dubey, Kevin Duh

Phil Edmonds, Markus Egg, Patrick Ehlen, Andreas Eisele, Jacob Eisenstein, Michael Elhadad, Micha Elsner, Katrin Erk, Gunes Erkan, David Evans, Stefan Evert

Yi Fang, Afsaneh Fazly, Ronen Feldman, Christiane Fellbaum, Raul Fernandez, Elena Filatova, Jenny Finkel, Michael Fleischman, Dan Flickinger, Radu Florian, Katherine Forbes, Eric Fosler-Lussier, Frederik Fouvry, Nissim Francez, Robert Frank, Alex Fraser, Bob Frederking, Marjorie Freedman, Dayne Freitag, Junichi Fukumoto

Evgeniy Gabrilovich, Robert Gaizauskas, Michael Gamon, Sudeep Gandhe, Yuqing Gao, Claire Gardent, Ulrich Germann, Roxana Girju, Natalie Glance, Oren Glickman, Amir Globerson, Yoav Goldberg, Ayelet Goldstein, Jade Goldstein, Sharon Goldwater, Gregory Grefenstette, Thomas Griffiths, Ralph Grishman, Iryna Gurevych, Joakim Gustafson

Stephanie Haas, Nizar Habash, Aria Haghighi, Tom Hain, Dilek Hakkani-Tur, Keith Hall, Sanda Harabagiu, Donna Harman, Sasa Hasan, Timothy Hazen, Daqing He, Xiaodong He, Mary Hearne, Marti Hearst, Ulrich Heid, James Henderson, Ulf Hermjakob, Andrew Hickl, Julia Hirschberg, Lynette Hirschman, Graeme Hirst, Julia Hockenmaier, Mark Hopkins, Veronique Hoste, Eduard Hovy, Churen Huang, Liang Huang, Sarmad Hussain, Bouke Huurnink, Mei-Yuh Hwang

Nancy Ide, Diana Inkpen, Kentaro Inui, Mitsuru Ishizuka, Abe Ittycheriah

Jagadeesh Jagarlamudi, Martin Jansche, Mark Johnson, Rie Johnson, Kristiina Jokinen, Gareth Jones, Rosie Jones, Aravind Joshi

Laura Kallmeyer, Nanda Kambhatla, Hiroshi Kanayama, Noriko Kando, Damianos Karakos, Nikiforos Karamanis, Hideki Kashioka, Yasuhiro Katagiri, Rohit Kate, Tsuneaki Kato, Boris Katz, Tatsuya Kawahara, Junichi Kazama, Bill Keler, Frank Keller, Charles Kemp, Andre Kempe, Stanley Yong Wai Keong, Sharam Khadivi, Mahboob Khalid, Rodger Kibble, Bernd Kiefer, Adam Kilgarriff, Chunyu Kit, Dan Klein, Kevin Knight, Alistair Knott, Philipp Koehn, Rob Koeling, Alexander Koller, Terry Koo, Moshe Koppel, Anna Korhonen, Kimmo Koskenniemi, Emiel Kraemer, Geert-Jan Kruijff, Yuval Krymlowski, Sandra Kuebler, Marco Kuhlmann, Jonas Kuhn, Seth Kulick, Shankar Kumar, A Kumaran, June-Jei Kuo, Sadao Kurohashi

Philippe Langlais, Mirella Lapata, Alex Lascarides, Alberto Lavelli, Alon Lavie, Victor Lavrenko, Alan Lee, Gary Lee, Lillian Lee, Yoong Keok Lee, Xin Lei, Gregor Leusch, Lori Levin, Hang Li, Jianguo Li, Qing Li, Xiaolong Li, Xiaoyan Li, Jimmy Lin, Krister Lindén, Lucian Lita, Ken Litkowski, Diane Litman, Bing Liu, Qun Liu, Tie-Yan Liu, Yang Liu, Karen Livescu, Andrei Ljolje, Adam Lopez, Yajuan Lu, Anke Lüdeling, Xiaoqiang Luo

Brian Mak, Rob Malouf, Inderjeet Mani, Gideon Mann, Daniel Marcu, Lluís Màrquez, Brandeis Marshall, Maria Antònia Martí, James Martin, Jean-Claude Martin, David Martínez, Gregory Marton, Mstislav Maslennikov, Tomoko Matsui, Yuji Matsumoto, Evgeny Matusov, Arne Mauser, Jonathan May, Mark Maybury, Diana McCarthy, Mark McConnville, Kathleen McCoy, Ryan McDonald, Tony Mcenry, Chris Mellish, Helen Meng, Paola Merlo, Detmar Meurers, Rada Mihalcea, Brian Milch, Eleni Miltsakaki, David Mimno, Wolfgang Minker, Einat Minkov, Gilad Mishne, Dipti Misra, Teruko Mitamura, Mandar Mitra, Vibhu Mittal, Yusuke Miyao, Noboru Miyazaki, Sien Moens, Saif Mohammad, Rajat Mohanty, Dan Moldovan, Diego Mollá, Christian Monson, Christof Monz, Raymond Mooney, Bob Moore, Glyn Morrill, Alessandro Moschitti, Karin Müller, Dragos Munteanu, Smaranda Muresan, Reinhard Muskens, Sung Hyon Myaeng

Masaaki Nagata, Mikio Nakano, Yukiko Nakano, Vivi Nastase, Roberto Navigli, Mark-Jan Nederhof, Ani Nenkova, John Nerbonne, Günter Neumann, Hermann Ney, Hwee Tou Ng, Vincent Ng, Patrick Nguyen, Jian-Yun Nie, Zaiqing Nie, Takashi Ninomiya, Malvina Nissim, Cheng Niu, Joakim Nivre, Chikashi Nobata, Elena Not, Adrian Novischi

Jon Oberlander, Franz Och, Stephan Oepen, Kemal Oflazer, Manabu Okumura, Miles Osborne, Jahna Otterbacher

Sebastian Pado, Tim Paek, Martha Palmer, Bo Pang, Cecile Paris, Marius Pasca, Rebecca Passonneau, Jon Patrick, Siddharth Patwardhan, Michael Paul, Adam Pease, Ted Pedersen, Catherine Pelachaud, Anselmo Peñas, Gerald Penn, Wim Peters, Paul Piwek, Massimo Poesio, Octavian Popescu, Andrei Popescu-Belis, Maja Popovic, Chris Potts, Richard Power, Sameer Pradhan, John Prager, Rashmi Prasad, Detlef Prescher, Stephen Pulman, Amruta Purandare, James Pustejovsky

Long Qiu, Yan Qu, Chris Quirk

Hema Raghavan, Bhuvana Ramabhadran, Ganesh Ramakrishnan, Owen Rambow, Lance Ramshaw, Deepak Ravichandran, Ehud Reiter, Norbert Reithinger, Philip Resnik, Giuseppe Riccardi, Stefan Riezler, German Rigau, Ellen Riloff, Hae-Chang Rim, Fabio Rinaldi, Brian Roark, James Rogers, Maribel Romero, Barbara Rosario, Dan Roth, Victoria Rubin

Kenji Sagae, Horacio Saggion, Tetsuya Sakai, Joan A. Sánchez, Mark Sanderson, Murat Saraclar, Anoop Sarkar, Shudeshna Sarkar, Yutaka Sasaki, Giorgio Satta, Jan Schehl, Michael Schiehlen, Anne Schiller, David Schlangen, Judith Schlesinger, Helmut Schmid, Marc Schroeder, Hinrich Schütze, Holger Schwenk, Donia Scott, Satoshi Sekine, Mike Seltzer, Vijay Shanker, Libin Shen, Akira Shimazu, Luo Si, Advait Siddharthan, Melanie Siegel, Khalil Simaan, Michel Simard, David Smith, Rion Snow, Benjamin Snyder, Stephen Soderland, Anders Sjøgaard, Swapna Somasundaran, David Sonntag, Jennifer Spenader, Caroline Sporleder, Richard Sproat, Manfred Stede, Mark Steedman, Amanda Stent, Mark Stevenson, Suzanne Stevenson, Nicola Stokes, Matthew Stone, Veselin Stoyanov, Carlo Strapparava, Michael Strube, Tomek Strzalkowski, Jian Su, Keh-Yih Su, Eiichiro Sumita, Jian-Tao Sun, Richard Sutcliffe, Charles Sutton, Idan Szpektor

Maite Taboada, John Tait, Hiroya Takamura, David Talbot, Pasi Tapanainen, Joel Tetreault, Mariet Theune, Vu Thuy, Jörg Tiedemann, Christoph Tillmann, Roberto Togneri, Takenobu Tokunaga, Kristina Toutanova, David Traum, Benjamin Tsou, Hajime Tsukada, Yoshimasa Tsuruoka, Gokhan Tur, Peter Turney

Raghavendra Udupa, Nicola Ueffing, Masao Utiyama

Antal van den Bosch, Josef van Genabith, Hans van Halteren, Lucy Vanderwende, Tony Veale, Sriram Venkatapathy, Ashish Venugopal, Marc Verhagen, Paola Verlardi, Yannick Versley, Renata Vieira, David Vilar, Piek Vossen, Atro Voutilainen

Joachim Wagner, Marilyn Walker, Michael Walsh, Xiaojun Wan, Haifeng Wang, Wei Wang, Bonnie Webber, Wouter Weerkamp, Ben Wellner, Fuiliang Weng, Michael White, Richard Wicentowski, Yorick Wilks, Theresa Wilson, Shuly Wintner, Yuk Wah Wong, Johan Wouters, Dekai Wu

Fei Xia, Jingfang Xu, Peng Xu

Atsushi Yamada, Kazuhide Yamamoto, Xiaofeng Yang, Alexander Yates, Shiren Ye, Scott Wen-tau Yih, Clem Yu, Deniz Yuret

Dmitry Zaykovskiy, Dmitry Zelenko, Luke Zettlemoyer, ChengXiang Zhai, Hao Zhang, Min Zhang, Rong Zhang, Tong Zhang, Yue Zhang, Jerry Zhu, Andreas Zollmann, Chengqing Zong, Ingrid Zukerman, Pierre Zweigenbaum

# Conference Program

## Monday, June 16, 2008

9:00–9:10      Opening Session

9:10–10:10    Invited Talk: Marc Swerts, *Facial Expressions in Human-Human and Human-Machine Interactions*

10:10–10:40    **Break**

### Session 1A: Information Extraction 1

10:40–11:05    *Mining Wiki Resources for Multilingual Named Entity Recognition*  
Alexander E. Richman and Patrick Schone

11:05–11:30    *Distributional Identification of Non-Referential Pronouns*  
Shane Bergsma, Dekang Lin and Randy Goebel

11:30–11:55    *Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs*  
Marius Paşca and Benjamin Van Durme

11:55–12:20    *The Tradeoffs Between Open and Traditional Relation Extraction*  
Michele Banko and Oren Etzioni

### Session 1B: Language Resources and Evaluation

10:40–11:05    *PDT 2.0 Requirements on a Query Language*  
Jiří Mírovský

11:05–11:30    *Task-oriented Evaluation of Syntactic Parsers and Their Representations*  
Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki and Jun'ichi Tsujii

11:30–11:55    *MAXSIM: A Maximum Similarity Metric for Machine Translation Evaluation*  
Yee Seng Chan and Hwee Tou Ng

11:55–12:20    *Contradictions and Justifications: Extensions to the Textual Entailment Task*  
Ellen M. Voorhees

**Monday, June 16, 2008 (continued)**

**Session 1C: Machine Translation 1**

- 10:40–11:05 *Cohesive Phrase-Based Decoding for Statistical Machine Translation*  
Colin Cherry
- 11:05–11:30 *Phrase Table Training for Precision and Recall: What Makes a Good Phrase and a Good Phrase Pair?*  
Yonggang Deng, Jia Xu and Yuqing Gao
- 11:30–11:55 *Measure Word Generation for English-Chinese SMT Systems*  
Dongdong Zhang, Mu Li, Nan Duan, Chi-Ho Li and Ming Zhou
- 11:55–12:20 *Bayesian Learning of Non-Compositional Phrases with Synchronous Parsing*  
Hao Zhang, Chris Quirk, Robert C. Moore and Daniel Gildea

**Session 1D: Speech Processing**

- 10:40–11:05 *Applying a Grammar-Based Language Model to a Simplified Broadcast-News Transcription Task*  
Tobias Kaufmann and Beat Pfister
- 11:05–11:30 *Automatic Editing in a Back-End Speech-to-Text System*  
Maximilian Bisani, Paul Vozila, Olivier Divay and Jeff Adams
- 11:30–11:55 *Grounded Language Modeling for Automatic Speech Recognition of Sports Video*  
Michael Fleischman and Deb Roy
- 11:55–12:20 *Lexicalized Phonotactic Word Segmentation*  
Margaret M. Fleck
- 12:20–2:00 **Lunch**

**Monday, June 16, 2008 (continued)**

**Session 2A: Information Retrieval 1**

- 2:00–2:25 *A Re-examination of Query Expansion Using Lexical Resources*  
Hui Fang
- 2:25–2:50 *Selecting Query Term Alternations for Web Search by Exploiting Query Contexts*  
Guihong Cao, Stephen Robertson and Jian-Yun Nie
- 2:50–3:15 *Searching Questions by Identifying Question Topic and Question Focus*  
Huizhong Duan, Yunbo Cao, Chin-Yew Lin and Yong Yu

**Session 2B: Language Generation**

- 2:00–2:25 *Trainable Generation of Big-Five Personality Styles through Data-Driven Parameter Estimation*  
François Mairesse and Marilyn Walker
- 2:25–2:50 *Correcting Misuse of Verb Forms*  
John Lee and Stephanie Seneff
- 2:50–3:15 *Hypertagging: Supertagging for Surface Realization with CCG*  
Dominic Espinosa, Michael White and Dennis Mehay

**Session 2C: Machine Translation 2**

- 2:00–2:25 *Forest-Based Translation*  
Haitao Mi, Liang Huang and Qun Liu
- 2:25–2:50 *A Discriminative Latent Variable Model for Statistical Machine Translation*  
Phil Blunsom, Trevor Cohn and Miles Osborne
- 2:50–3:15 *Efficient Multi-Pass Decoding for Synchronous Context Free Grammars*  
Hao Zhang and Daniel Gildea

**Monday, June 16, 2008 (continued)**

**Session 2D: Semantics 1**

- 2:00–2:25 *Regular Tree Grammars as a Formalism for Scope Underspecification*  
Alexander Koller, Michaela Regneri and Stefan Thater
- 2:25–2:50 *Classification of Semantic Relationships between Nominals Using Pattern Clusters*  
Dmitry Davidov and Ari Rappoport
- 2:50–3:15 *Vector-based Models of Semantic Composition*  
Jeff Mitchell and Mirella Lapata
- 3:15–3:45 **Break**

**Session 3A: Information Extraction 2**

- 3:45–4:10 *Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition*  
Andrew Arnold, Ramesh Nallapati and William W. Cohen
- 4:10–4:35 *Refining Event Extraction through Cross-Document Inference*  
Heng Ji and Ralph Grishman
- 4:35–5:00 *Learning Document-Level Semantic Properties from Free-Text Annotations*  
S.R.K. Branavan, Harr Chen, Jacob Eisenstein and Regina Barzilay
- 5:00–5:25 *Automatic Image Annotation Using Auxiliary Text Information*  
Yansong Feng and Mirella Lapata



**Monday, June 16, 2008 (continued)**

**Session 3B: Sentiment Analysis**

- 3:45–4:10 *Hedge Classification in Biomedical Texts with a Weakly Supervised Selection of Keywords*  
György Szarvas
- 4:10–4:35 *When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging*  
Alina Andreevskaia and Sabine Bergler
- 4:35–5:00 *A Generic Sentence Trimmer with CRFs*  
Tadashi Nomoto
- 5:00–5:25 *A Joint Model of Text and Aspect Ratings for Sentiment Summarization*  
Ivan Titov and Ryan McDonald

**Session 3C: Syntax and Parsing 1**

- 3:45–4:10 *Improving Parsing and PP Attachment Performance with Sense Information*  
Eneko Agirre, Timothy Baldwin and David Martinez
- 4:10–4:35 *A Logical Basis for the D Combinator and Normal Form in CCG*  
Frederick Hoyt and Jason Baldridge
- 4:35–5:00 *Parsing Noun Phrase Structure with CCG*  
David Vadas and James R. Curran
- 5:00–5:25 *Sentence Simplification for Semantic Role Labeling*  
David Vickrey and Daphne Koller

**Monday, June 16, 2008 (continued)**

**Session 3D: Student Research Workshop**

- 3:45–4:10 *A Supervised Learning Approach to Automatic Synonym Identification Based on Distributional Features*  
Masato Hagiwara
- 4:10–4:35 *An Integrated Architecture for Generating Parenthetical Constructions*  
Eva Banik
- 4:35–5:00 *Inferring Activity Time in News through Event Modeling*  
Vladimir Eidelman
- 5:00–5:25 *Combining Source and Target Language Information for Name Tagging of Machine Translation Output*  
Shasha Liao
- 5:25–5:50 *A Re-examination on Features in Regression Based Approach to Automatic MT Evaluation*  
Shuqi Sun, Yin Chen and Jufeng Li
- 5:25–6:00 **Break**
- 6:00–8:30 **Poster and Demo Session**

**Long Paper Posters**

*Summarizing Emails with Conversational Cohesion and Subjectivity*  
Giuseppe Carenini, Raymond T. Ng and Xiaodong Zhou

*Ad Hoc Treebank Structures*  
Markus Dickinson

*A Single Generative Model for Joint Morphological Segmentation and Syntactic Parsing*  
Yoav Goldberg and Reut Tsarfaty

*Which Words Are Hard to Recognize? Prosodic, Lexical, and Disfluency Factors that Increase ASR Error Rates*  
Sharon Goldwater, Dan Jurafsky and Christopher D. Manning

*Name Translation in Statistical Machine Translation - Learning When to Transliterate*  
Ulf Hermjakob, Kevin Knight and Hal Daumé III

**Monday, June 16, 2008 (continued)**

*Using Adaptor Grammars to Identify Synergies in the Unsupervised Acquisition of Linguistic Structure*

Mark Johnson

*Inducing Gazetteers for Named Entity Recognition by Large-Scale Clustering of Dependency Relations*

Jun'ichi Kazama and Kentaro Torisawa

*Evaluating Roget's Thesauri*

Alistair Kennedy and Stan Szpakowicz

*Unsupervised Translation Induction for Chinese Abbreviations using Monolingual Corpora*

Zhifei Li and David Yarowsky

*Which Are the Best Features for Automatic Verb Classification*

Jianguo Li and Chris Brew

*Collecting a Why-Question Corpus for Development and Evaluation of an Automatic QA-System*

Joanna Mrozinski, Edward Whittaker and Sadaoki Furui

*Solving Relational Similarity Problems Using the Web as a Corpus*

Preslav Nakov and Marti A. Hearst

*Combining Speech Retrieval Results with Generalized Additive Models*

J. Scott Olsson and Douglas W. Oard

*A Critical Reassessment of Evaluation Baselines for Speech Summarization*

Gerald Penn and Xiaodan Zhu

*Intensional Summaries as Cooperative Responses in Dialogue: Automation and Evaluation*

Joseph Polifroni and Marilyn Walker

*Word Clustering and Word Selection Based Feature Reduction for MaxEnt Based Hindi NER*

Sujan Kumar Saha, Pabitra Mitra and Sudeshna Sarkar

*Combining EM Training and the MDL Principle for an Automatic Verb Classification Incorporating Selectional Preferences*

Sabine Schulte im Walde, Christian Hying, Christian Scheible and Helmut Schmid

**Monday, June 16, 2008 (continued)**

*Randomized Language Models via Perfect Hash Functions*

David Talbot and Thorsten Brants

*Applying Morphology Generation Models to Machine Translation*

Kristina Toutanova, Hisami Suzuki and Achim Ruopp

*Multilingual Harvesting of Cross-Cultural Stereotypes*

Tony Veale, Yanfen Hao and Guofu Li

*Semi-Supervised Convex Training for Dependency Parsing*

Qin Iris Wang, Dale Schuurmans and Dekang Lin

*Chinese-English Backward Transliteration Assisted with Mining Monolingual Web Pages*

Fan Yang, Jun Zhao, Bo Zou, Kang Liu and Feifan Liu

*Robustness and Generalization of Role Sets: PropBank vs. VerbNet*

Beñat Zepirain, Eneko Agirre and Lluís Màrquez

*A Tree Sequence Alignment-based Tree-to-Tree Translation Model*

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan and Sheng Li

#### **Short Paper Posters**

*Language Dynamics and Capitalization using Maximum Entropy*

Fernando Batista, Nuno Mamede and Isabel Trancoso

*Surprising Parser Actions and Reading Difficulty*

Marisa Ferrara Boston, John T. Hale, Reinhold Kliegl and Shravan Vasishth

*Improving the Performance of the Random Walk Model for Answering Complex Questions*

Yllias Chali and Shafiq Joty

*Dimensions of Subjectivity in Natural Language*

Wei Chen

*Extractive Summaries for Educational Science Content*

Sebastian de la Chica, Faisal Ahmad, James H. Martin and Tamara Sumner

**Monday, June 16, 2008 (continued)**

*Dialect Classification for Online Podcasts Fusing Acoustic and Language Based Structural and Semantic Information*

Rahul Chitturi and John Hansen

*The Complexity of Phrase Alignment Problems*

John DeNero and Dan Klein

*Novel Semantic Features for Verb Sense Disambiguation*

Dmitriy Dligach and Martha Palmer

*Icelandic Data Driven Part of Speech Tagging*

Mark Dredze and Joel Wallenberg

*Beyond Log-Linear Models: Boosted Minimum Error Rate Training for N-best Re-ranking*

Kevin Duh and Katrin Kirchhoff

*Coreference-inspired Coherence Modeling*

Micha Elsner and Eugene Charniak

*Enforcing Transitivity in Coreference Resolution*

Jenny Rose Finkel and Christopher D. Manning

*Simulating the Behaviour of Older versus Younger Users when Interacting with Spoken Dialogue Systems*

Kallirroi Georgila, Maria Wolters and Johanna Moore

*Active Sample Selection for Named Entity Transliteration*

Dan Goldwasser and Dan Roth

*Four Techniques for Online Handling of Out-of-Vocabulary Words in Arabic-English Statistical Machine Translation*

Nizar Habash

*Combined One Sense Disambiguation of Abbreviations*

Yaakov HaCohen-Kerner, Ariel Kass and Ariel Peretz

*Assessing the Costs of Sampling Methods in Active Learning for Annotation*

Robbie Haertel, Eric Ringger, Kevin Seppi, Carroll James and McClanahan Peter

*Blog Categorization Exploiting Domain Dictionary and Dynamically Estimated Domains of Unknown Words*

Chikara Hashimoto and Sadao Kurohashi

**Monday, June 16, 2008 (continued)**

*Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management*  
James Henderson and Oliver Lemon

*Recent Improvements in the CMU Large Scale Chinese-English SMT System*  
Almut Silja Hildebrand, Kay Rottmann, Mohamed Noamany, Quin Gao, Sanjika Hewavitharana, Nguyen Bach and Stephan Vogel

*Machine Translation System Combination using ITG-based Alignments*  
Damianos Karakos, Jason Eisner, Sanjeev Khudanpur and Markus Dreyer

*Dictionary Definitions based Homograph Identification using a Generative Hierarchical Model*  
Anagha Kulkarni and Jamie Callan

*A Novel Feature-based Approach to Chinese Entity Relation Extraction*  
Wenjie Li, Peng Zhang, Furu Wei, Yuexian Hou and Qin Lu

*Using Structural Information for Identifying Similar Chinese Characters*  
Chao-Lin Liu and Jen-Hsiang Lin

*You've Got Answers: Towards Personalized Models for Predicting Success in Community Question Answering*  
Yandong Liu and Eugene Agichtein

*Self-Training for Biomedical Parsing*  
David McClosky and Eugene Charniak

*A Unified Syntactic Model for Parsing Fluent and Disfluent Speech*  
Tim Miller and William Schuler

*The Good, the Bad, and the Unknown: Morphosyllabic Sentiment Tagging of Unseen Words*  
Karo Moilanen and Stephen Pulman

*Kernels on Linguistic Structures for Answer Extraction*  
Alessandro Moschitti and Silvia Quarteroni

*Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking*  
Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab and Cynthia Rudin

*Using Automatically Transcribed Dialogs to Learn User Models in a Spoken Dialog System*  
Umar Syed and Jason Williams

**Monday, June 16, 2008 (continued)**

*Robust Extraction of Named Entity Including Unfamiliar Word*  
Masatoshi Tsuchiya, Shinya Hida and Seichi Nakagawa

*In-Browser Summarisation: Generating Elaborative Summaries Biased Towards the Reading Context*  
Stephen Wan and Cécile Paris

*Lyric-based Song Sentiment Classification with Sentiment Vector Space Model*  
Yunqing Xia, Linlin Wang, Kam-Fai Wong and Mingxing Xu

*Mining Wikipedia Revision Histories for Improving Sentence Compression*  
Elif Yamangil and Rani Nelken

*Smoothing a Tera-word Language Model*  
Deniz Yuret

**Student Research Workshop Posters**

*The Role of Positive Feedback in Intelligent Tutoring Systems*  
Davide Fossati

*Arabic Language Modeling with Finite State Transducers*  
Ilana Heintz

*Impact of Initiative on Collaborative Problem Solving*  
Cynthia Kersey

*An Unsupervised Vector Approach to Biomedical Term Disambiguation: Integrating UMLS and Medline*  
Bridget McInnes

*A Subcategorization Acquisition System for French Verbs*  
Cédric Messiant

*Adaptive Language Modeling for Word Prediction*  
Keith Trnka

*A Hierarchical Approach to Encoding Medical Concepts for Clinical Notes*  
Yitao Zhang

**Monday, June 16, 2008 (continued)**

**Demonstrations**

*Demonstration of a POMDP Voice Dialer*

Jason Williams

*Generating Research Websites Using Summarisation Techniques*

Advaith Siddharthan and Ann Copestake

*BART: A Modular Toolkit for Coreference Resolution*

Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang and Alessandro Moschitti

*Demonstration of the UAM CorpusTool for Text and Image Annotation*

Mick O'Donnell

*Interactive ASR Error Correction for Touchscreen Devices*

David Huggins-Daines and Alexander I. Rudnicky

*Yawat: Yet Another Word Alignment Tool*

Ulrich Germann

*SIDE: The Summarization Integrated Development Environment*

Moonyoung Kang, Sourish Chaudhuri, Mahesh Joshi and Carolyn P. Rosé

*ModelTalker Voice Recorder—An Interface System for Recording a Corpus of Speech for Synthesis*

Debra Yarrington, John Gray, Chris Pennington, H. Timothy Bunnell, Allegra Cornaglia, Jason Lilley, Kyoko Nagao and James Polikoff

*The QuALiM Question Answering Demo: Supplementing Answers with Paragraphs drawn from Wikipedia*

Michael Kaiser



**Tuesday, June 17, 2008**

**Session: Outstanding Paper Award Presentations**

9:00–9:10 Presentation of Awards

9:10–9:35 *Automatic Syllabification with Structured SVMs for Letter-to-Phoneme Conversion*  
Susan Bartlett, Grzegorz Kondrak and Colin Cherry

9:35–10:00 *A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model*  
Libin Shen, Jinxi Xu and Ralph Weischedel

10:00–10:25 *Forest Reranking: Discriminative Parsing with Non-Local Features*  
Liang Huang

10:25–10:40 *Event Matching Using the Transitive Closure of Dependency Relations*  
Daniel M. Bikel and Vittorio Castelli

10:40–11:10 **Break**

**Session 4A: Syntax and Parsing 2**

11:10–11:35 *Simple Semi-supervised Dependency Parsing*  
Terry Koo, Xavier Carreras and Michael Collins

11:35–12:00 *Optimal  $k$ -arization of Synchronous Tree-Adjoining Grammar*  
Rebecca Nesson, Giorgio Satta and Stuart M. Shieber

12:00–12:25 *Enhancing Performance of Lexicalised Grammars*  
Rebecca Dridan, Valia Kordoni and Jeremy Nicholson

**Tuesday, June 17, 2008 (continued)**

**Session 4B: Dialogue**

- 11:10–11:35 *Assessing Dialog System User Simulation Evaluation Measures Using Human Judges*  
Hua Ai and Diane J. Litman
- 11:35–12:00 *Robust Dialog Management with N-Best Hypotheses Using Dialog Examples and Agenda*  
Cheongjae Lee, Sangkeun Jung and Gary Geunbae Lee
- 12:00–12:25 *Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz Data: Bootstrapping and Evaluation*  
Verena Rieser and Oliver Lemon

**Session 4C: Machine Learning 2**

- 11:10–11:35 *Phrase Chunking Using Entropy Guided Transformation Learning*  
Ruy Luiz Milidiú, Cícero Nogueira dos Santos and Julio C. Duarte
- 11:35–12:00 *Learning Bigrams from Unigrams*  
Xiaojin Zhu, Andrew B. Goldberg, Michael Rabbat and Robert Nowak
- 12:00–12:25 *Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data*  
Jun Suzuki and Hideki Isozaki

**Session 4D: Semantics 2**

- 11:10–11:35 *Large Scale Acquisition of Paraphrases for Learning Surface Patterns*  
Rahul Bhagat and Deepak Ravichandran
- 11:35–12:00 *Contextual Preferences*  
Idan Szpektor, Ido Dagan, Roy Bar-Haim and Jacob Goldberger
- 12:00–12:25 *Unsupervised Discovery of Generic Relationships Using Pattern Clusters and its Evaluation by Automatically Generated SAT Analogy Questions*  
Dmitry Davidov and Ari Rappoport
- 12:25–2:00 **Lunch**

**Tuesday, June 17, 2008 (continued)**

**Session 5A: Short Papers 1 (Machine Translation)**

- 2:00–2:15 *A Linguistically Annotated Reordering Model for BTG-based Statistical Machine Translation*  
Deyi Xiong, Min Zhang, Aiti Aw and Haizhou Li
- 2:15–2:30 *Segmentation for English-to-Arabic Statistical Machine Translation*  
Ibrahim Badr, Rabih Zbib and James Glass
- 2:30–2:45 *Exploiting N-best Hypotheses for SMT Self-Enhancement*  
Boxing Chen, Min Zhang, Aiti Aw and Haizhou Li
- 2:45–3:00 *Partial Matching Strategy for Phrase-based Statistical Machine Translation*  
Zhongjun He, Qun Liu and Shouxun Lin

**Session 5B: Short Papers 2 (Speech)**

- 2:00–2:15 No presentation
- 2:15–2:30 *Unsupervised Learning of Acoustic Sub-word Units*  
Balakrishnan Varadarajan, Sanjeev Khudanpur and Emmanuel Dupoux
- 2:30–2:45 *High Frequency Word Entrainment in Spoken Dialogue*  
Ani Nenkova, Agustín Gravano and Julia Hirschberg
- 2:45–3:00 *Distributed Listening: A Parallel Processing Approach to Automatic Speech Recognition*  
Yolanda McMillian and Juan Gilbert

**Tuesday, June 17, 2008 (continued)**

**Session 5C: Short Papers 3 (Semantics)**

- 2:00–2:15 *Learning Semantic Links from a Corpus of Parallel Temporal and Causal Relations*  
Steven Bethard and James H. Martin
- 2:15–2:30 *Evolving New Lexical Association Measures Using Genetic Programming*  
Jan Šnajder, Bojana Dalbelo Bašič, Saša Petrovič and Ivan Sikirič
- 2:30–2:45 *Semantic Types of Some Generic Relation Arguments: Detection and Evaluation*  
Sophia Katrenko and Pieter Adriaans
- 2:45–3:00 *Mapping between Compositional Semantic Representations and Lexical Semantic Resources: Towards Accurate Deep Semantic Parsing*  
Sergio Roa, Valia Kordoni and Yi Zhang

**Session 5D: Short Papers 4 (Generation/Summarization)**

- 2:00–2:15 *Query-based Sentence Fusion is Better Defined and Leads to More Preferred Results than Generic Sentence Fusion*  
Emiel Kraemer, Erwin Marsi and Paul van Pelt
- 2:15–2:30 *Intrinsic vs. Extrinsic Evaluation Measures for Referring Expression Generation*  
Anja Belz and Albert Gatt
- 2:30–2:45 *Correlation between ROUGE and Human Evaluation of Extractive Meeting Summaries*  
Feifan Liu and Yang Liu
- 2:45–3:00 *FastSum: Fast and Accurate Query-based Multi-document Summarization*  
Frank Schilder and Ravikumar Kondadadi
- 3:00–3:15 **Break**

**Tuesday, June 17, 2008 (continued)**

**Session 5E: Short Papers 1 (Syntax)**

- 3:15–3:30     *Construct State Modification in the Arabic Treebank*  
Ryan Gabbard and Seth Kulick
- 3:30–3:45     *Unlexicalised Hidden Variable Models of Split Dependency Grammars*  
Gabriele Antonio Musillo and Paola Merlo
- 3:45–4:00     *Computing Confidence Scores for All Sub Parse Trees*  
Feng Lin and Fuliang Weng
- 4:00–4:15     *Adapting a WSJ-Trained Parser to Grammatically Noisy Text*  
Jennifer Foster, Joachim Wagner and Josef van Genabith

**Session 5F: Short Papers 2 (Dialog/Statistical Methods)**

- 3:15–3:30     *Enriching Spoken Language Translation with Dialog Acts*  
Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore and Shrikanth Narayanan
- 3:30–3:45     *Speakers' Intention Prediction Using Statistics of Multi-level Features in a Schedule Management Domain*  
Donghyun Kim, Hyunjung Lee, Choong-Nyoung Seon, Harksoo Kim and Jungyun Seo
- 3:45–4:00     *Active Learning with Confidence*  
Mark Dredze and Koby Crammer
- 4:00–4:15     *splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications*  
Yoav Goldberg and Michael Elhadad

**Tuesday, June 17, 2008 (continued)**

**Session 5G: Short Papers 3 (Semantics/Phonology)**

- 3:15–3:30 *Extracting a Representation from Text for Semantic Analysis*  
Rodney D. Nielsen, Wayne Ward, James H. Martin and Martha Palmer
- 3:30–3:45 *Efficient Processing of Underspecified Discourse Representations*  
Michaela Regneri, Markus Egg and Alexander Koller
- 3:45–4:00 *Choosing Sense Distinctions for WSD: Psycholinguistic Evidence*  
Susan Windisch Brown
- 4:00–4:15 *Decompounding query keywords from compounding languages*  
Enrique Alfonseca, Slaven Bilac and Stefan Pharies

**Session 5H: Short Papers 4 (Information Retrieval/Sentiment Analysis)**

- 3:15–3:30 *Multi-domain Sentiment Classification*  
Shoushan Li and Chengqing Zong
- 3:30–3:45 *Evaluating Word Prediction: Framing Keystroke Savings*  
Keith Trnka and Kathleen McCoy
- 3:45–4:00 *Pairwise Document Similarity in Large Collections with MapReduce*  
Tamer Elsayed, Jimmy Lin and Douglas Oard
- 4:00–4:15 *Text Segmentation with LDA-Based Fisher Kernel*  
Qi Sun, Runxin Li, Dingsheng Luo and Xihong Wu
- 4:15–4:45 **Break**

**Tuesday, June 17, 2008 (continued)**

**Session 6A: Question Answering**

- 4:45–5:10 *Improving Search Results Quality by Customizing Summary Lengths*  
Michael Kaisser, Marti A. Hearst and John B. Lowe
- 5:10–5:35 *Using Conditional Random Fields to Extract Contexts and Answers of Questions from Online Forums*  
Shilin Ding, Gao Cong, Chin-Yew Lin and Xiaoyan Zhu
- 5:35–6:00 *Learning to Rank Answers on Large Online QA Collections*  
Mihai Surdeanu, Massimiliano Ciaramita and Hugo Zaragoza

**Session 6B: Phonology, Morphology 1**

- 4:45–5:10 *Unsupervised Lexicon-Based Resolution of Unknown Words for Full Morphological Analysis*  
Meni Adler, Yoav Goldberg, David Gabay and Michael Elhadad
- 5:10–5:35 *Unsupervised Multilingual Learning for Morphological Segmentation*  
Benjamin Snyder and Regina Barzilay
- 5:35–6:00 *EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start)*  
Yoav Goldberg, Meni Adler and Michael Elhadad

**Session 6C: Machine Translation 3**

- 4:45–5:10 *Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation*  
Jakob Uszkoreit and Thorsten Brants
- 5:10–5:35 *Enriching Morphologically Poor Languages for Statistical Machine Translation*  
Eleftherios Avramidis and Philipp Koehn
- 5:35–6:00 *Learning Bilingual Lexicons from Monolingual Corpora*  
Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick and Dan Klein

**Tuesday, June 17, 2008 (continued)**

**Session 6D: Semantics 3**

4:45–5:10 *Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora*  
Shiqi Zhao, Haifeng Wang, Ting Liu and Sheng Li

5:10–5:35 *Unsupervised Learning of Narrative Event Chains*  
Nathanael Chambers and Dan Jurafsky

5:35–6:00 *Semantic Role Labeling Systems for Arabic using Kernel Methods*  
Mona Diab, Alessandro Moschitti and Daniele Pighin

7:00–11:00 **Banquet**

**Wednesday, June 18, 2008**

9:00–10:00 Invited Talk: Susan Dumais, *Supporting Searchers in Searching*

10:00–10:30 **Break**

**Session 7A: Summarization**

10:30–10:55 *An Unsupervised Approach to Biography Production Using Wikipedia*  
Fadi Biadisy, Julia Hirschberg and Elena Filatova

10:55–11:20 *Generating Impact-Based Summaries for Scientific Literature*  
Qiaozhu Mei and ChengXiang Zhai

11:20–11:45 *Can You Summarize This? Identifying Correlates of Input Difficulty for Multi-Document Summarization*  
Ani Nenkova and Annie Louis



**Wednesday, June 18, 2008 (continued)**

**Session 7B: Discourse and Pragmatics**

- 10:30–10:55 *You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement*  
Micha Elsner and Eugene Charniak
- 10:55–11:20 *An Entity-Mention Model for Coreference Resolution with Inductive Logic Programming*  
Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, Ting Liu and Sheng Li
- 11:20–11:45 *Gestural Cohesion for Topic Segmentation*  
Jacob Eisenstein, Regina Barzilay and Randall Davis

**Session 7C: Machine Learning 2**

- 10:30–10:55 *Multi-Task Active Learning for Linguistic Annotations*  
Roi Reichart, Katrin Tomanek, Udo Hahn and Ari Rappoport
- 10:55–11:20 *Generalized Expectation Criteria for Semi-Supervised Learning of Conditional Random Fields*  
Gideon S. Mann and Andrew McCallum
- 11:20–11:45 *Analyzing the Errors of Unsupervised Learning*  
Percy Liang and Dan Klein

**Session 7D: Phonology, Morphology 2**

- 10:30–10:55 *Joint Word Segmentation and POS Tagging Using a Single Perceptron*  
Yue Zhang and Stephen Clark
- 10:55–11:20 *A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging*  
Wenbin Jiang, Liang Huang, Qun Liu and Yajuan Lü
- 11:20–11:45 *Joint Processing and Discriminative Training for Letter-to-Phoneme Conversion*  
Sittichai Jiampojarn, Colin Cherry and Grzegorz Kondrak
- 11:45–1:15 **ACL Business Meeting**
- 1:15–2:30 **Lunch**

**Wednesday, June 18, 2008 (continued)**

**Session 8A: Information Retrieval 2**

- 2:30–2:55 *A Probabilistic Model for Fine-Grained Expert Search*  
Shenghua Bao, Huizhong Duan, Qi Zhou, Miao Xiong, Yunbo Cao and Yong Yu
- 2:55–3:20 *Credibility Improves Topical Blog Post Retrieval*  
Wouter Weerkamp and Maarten de Rijke
- 3:20–3:45 *Linguistically Motivated Features for Enhanced Back-of-the-Book Indexing*  
Andras Csomai and Rada Mihalcea
- 3:45–4:10 *Resolving Personal Names in Email Using Context Expansion*  
Tamer Elsayed, Douglas W. Oard and Galileo Namata

**Session 8B: Syntax and Parsing 3**

- 2:30–2:55 *Integrating Graph-Based and Transition-Based Dependency Parsers*  
Joakim Nivre and Ryan McDonald
- 2:55–3:20 *Efficient, Feature-based, Conditional Random Field Parsing*  
Jenny Rose Finkel, Alex Kleeman and Christopher D. Manning
- 3:20–3:45 *A Deductive Approach to Dependency Parsing*  
Carlos Gómez-Rodríguez, John Carroll and David Weir
- 3:45–4:10 *Evaluating a Crosslinguistic Grammar Resource: A Case Study of Wambaya*  
Emily M. Bender

**Wednesday, June 18, 2008 (continued)**

**Session 8C: Machine Translation 2**

- 2:30–2:55 *Better Alignments = Better Translations?*  
Kuzman Ganchev, João V. Graça and Ben Taskar
- 2:55–3:20 *Mining Parenthetical Translations from the Web by Word Alignment*  
Dekang Lin, Shaojun Zhao, Benjamin Van Durme and Marius Paşca
- 3:20–3:45 *Soft Syntactic Constraints for Hierarchical Phrased-Based Translation*  
Yuval Marton and Philip Resnik
- 3:45–4:10 *Generalizing Word Lattice Translation*  
Christopher Dyer, Smaranda Muresan and Philip Resnik

**Session 8D: Semantics 4**

- 2:30–2:55 *Combining Multiple Resources to Improve SMT-based Paraphrasing Model*  
Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu and Sheng Li
- 2:55–3:20 *Extraction of Entailed Semantic Relations Through Syntax-Based Comma Resolution*  
Vivek Srikumar, Roi Reichart, Mark Sammons, Ari Rappoport and Dan Roth
- 3:20–3:45 *Finding Contradictions in Text*  
Marie-Catherine de Marneffe, Anna N. Rafferty and Christopher D. Manning
- 3:45–4:10 *Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs*  
Zornitsa Kozareva, Ellen Riloff and Eduard Hovy
- 4:10–4:40 **Break**
- 4:40–5:50 Lifetime Achievement Award Presentation
- 5:50–6:00 Closing Session



# Mining Wiki Resources for Multilingual Named Entity Recognition

**Alexander E. Richman**

Department of Defense  
Washington, DC 20310  
arichman@psualum.com

**Patrick Schone**

Department of Defense  
Fort George G. Meade, MD 20755  
pjschon@tycho.ncsc.mil

## Abstract

In this paper, we describe a system by which the multilingual characteristics of Wikipedia can be utilized to annotate a large corpus of text with Named Entity Recognition (NER) tags requiring minimal human intervention and no linguistic expertise. This process, though of value in languages for which resources exist, is particularly useful for less commonly taught languages. We show how the Wikipedia format can be used to identify possible named entities and discuss in detail the process by which we use the Category structure inherent to Wikipedia to determine the named entity type of a proposed entity. We further describe the methods by which English language data can be used to bootstrap the NER process in other languages. We demonstrate the system by using the generated corpus as training sets for a variant of BBN's Identifinder in French, Ukrainian, Spanish, Polish, Russian, and Portuguese, achieving overall F-scores as high as 84.7% on independent, human-annotated corpora, comparable to a system trained on up to 40,000 words of human-annotated newswire.

## 1 Introduction

Named Entity Recognition (NER) has long been a major task of natural language processing. Most of the research in the field has been restricted to a few languages and almost all methods require substantial linguistic expertise, whether creating a rule-based technique specific to a language or manually annotating a body of text to be used as a training set for a statistical engine or machine learning.

In this paper, we focus on using the multilingual Wikipedia (wikipedia.org) to automatically create an annotated corpus of text in any given language, with no linguistic expertise required on the part of the user at run-time (and only English knowledge

required during development). The expectation is that for any language in which Wikipedia is sufficiently well-developed, a usable set of training data can be obtained with minimal human intervention. As Wikipedia is constantly expanding, it follows that the derived models are continually improved and that increasingly many languages can be usefully modeled by this method.

In order to make sure that the process is as language-independent as possible, we declined to make use of any non-English linguistic resources outside of the Wikimedia domain (specifically, Wikipedia and the English language Wiktionary (en.wiktionary.org)). In particular, we did not use any semantic resources such as WordNet or part of speech taggers. We used our automatically annotated corpus along with an internally modified variant of BBN's Identifinder (Bikel et al., 1999), specifically modified to emphasize fast text processing, called "PhoenixIDF," to create several language models that could be tested outside of the Wikipedia framework. We built on top of an existing system, and left existing lists and tables intact. Depending on language, we evaluated our derived models against human or machine annotated data sets to test the system.

## 2 Wikipedia

### 2.1 Structure

Wikipedia is a multilingual, collaborative encyclopedia on the Web which is freely available for research purposes. As of October 2007, there were over 2 million articles in English, with versions available in 250 languages. This includes 30 languages with at least 50,000 articles and another 40 with at least 10,000 articles. Each language is available for download (download.wikimedia.org) in a text format suitable for inclusion in a database. For the remainder of this paper, we refer to this format.

Within Wikipedia, we take advantage of five major features:

- *Article links*, links from one article to another of the same language;
- *Category links*, links from an article to special “Category” pages;
- *Interwiki links*, links from an article to a presumably equivalent, article in another language;
- *Redirect pages*, short pages which often provide equivalent names for an entity; and
- *Disambiguation pages*, a page with little content that links to multiple similarly named articles.

The first three types are collectively referred to as *wikilinks*.

A typical sentence in the database format looks like the following:

“Nescopeck Creek is a [[tributary]] of the [[North Branch Susquehanna River]] in [[Luzerne County, Pennsylvania|Luzerne County]].”

The double bracket is used to signify wikilinks. In this snippet, there are three articles links to English language Wikipedia pages, titled “Tributary,” “North Branch Susquehanna River,” and “Luzerne County, Pennsylvania.” Notice that in the last link, the phrase preceding the vertical bar is the name of the article, while the following phrase is what is actually displayed to a visitor of the webpage.

Near the end of the same article, we find the following representations of Category links: [[Category:Luzerne County, Pennsylvania]], [[Category:Rivers of Pennsylvania]], {{Pennsylvania-geo-stub}}. The first two are direct links to Category pages. The third is a link to a Template, which (among other things) links the article to “Category:Pennsylvania geography stubs”. We will typically say that a given entity *belongs to* those categories to which it is linked in these ways.

The last major type of wikilink is the link between different languages. For example, in the Turkish language article “Kanuni Sultan Süleyman” one finds a set of links including [[en:Suleiman the Magnificent]] and [[ru:Сулейман I]]. These represent links to the English language article “Suleiman the Magnificent” and the Russian language article “Сулейман I.” In almost all cases, the articles linked in this manner represent articles on the same subject.

A redirect page is a short entry whose sole purpose is to direct a query to the proper page. There are a few reasons that redirect pages exist, but the primary purpose is exemplified by the fact that “USA” is an entry which redirects the user to the page entitled “United States.” That is, in the vast majority of cases, redirect pages provide another name for an entity.

A disambiguation page is a special article which contains little content but typically lists a number of entries which might be what the user was seeking. For instance, the page “Franklin” contains 70 links, including the singer “Aretha Franklin,” the town “Franklin, Virginia,” the “Franklin River” in Tasmania, and the cartoon character “Franklin (Peanuts).” Most disambiguation pages are in Category:Disambiguation or one of its subcategories.

## 2.2 Related Studies

Wikipedia has been the subject of a considerable amount of research in recent years including Gabrilovich and Markovitch (2007), Strube and Ponzetto (2006), Milne et al. (2006), Zesch et al. (2007), and Weale (2007). The most relevant to our work are Kazama and Torisawa (2007), Toral and Muñoz (2006), and Cucerzan (2007). More details follow, but it is worth noting that all known prior results are fundamentally monolingual, often developing algorithms that can be adapted to other languages pending availability of the appropriate semantic resource. In this paper, we emphasize the use of links between articles of different languages, specifically between English (the largest and best linked Wikipedia) and other languages.

Toral and Muñoz (2006) used Wikipedia to create lists of named entities. They used the first sentence of Wikipedia articles as likely definitions of the article titles, and used them to attempt to classify the titles as people, locations, organizations, or none. Unlike the method presented in this paper, their algorithm relied on WordNet (or an equivalent resource in another language). The authors noted that their results would need to pass a manual supervision step before being useful for the NER task, and thus did not evaluate their results in the context of a full NER system.

Similarly, Kazama and Torisawa (2007) used Wikipedia, particularly the first sentence of each article, to create lists of entities. Rather than building entity dictionaries associating words and

phrases to the classical NER tags (PERSON, LOCATION, etc.) they used a noun phrase following forms of the verb “to be” to derive a label. For example, they used the sentence “Franz Fischler ... is an Austrian politician” to associate the label “politician” to the surface form “Franz Fischler.” They proceeded to show that the dictionaries generated by their method are useful when integrated into an NER system. We note that their technique relies upon a part of speech tagger, and thus was not appropriate for inclusion as part of our non-English system.

Cucerzan (2007), by contrast to the above, used Wikipedia primarily for Named Entity Disambiguation, following the path of Bunescu and Paşca (2006). As in this paper, and unlike the above mentioned works, Cucerzan made use of the explicit Category information found within Wikipedia. In particular, Category and related list-derived data were key pieces of information used to differentiate between various meanings of an ambiguous surface form. Unlike in this paper, Cucerzan did not make use of the Category information to identify a given entity as a member of any particular class. We also note that the NER component was not the focus of the research, and was specific to the English language.

### 3 Training Data Generation

#### 3.1 Initial Set-up and Overview

Our approach to multilingual NER is to pull back the decision-making process to English whenever possible, so that we could apply some level of linguistic expertise. In particular, by focusing on only one language, we could take maximum advantage of the Category structure, something very difficult to do in the general multilingual case.

For computational feasibility, we downloaded various language Wikipedias and the English language Wiktionary in their text (.xml) format and stored each language as a table within a single MySQL database. We only stored the title, id number, and body (the portion between the <TEXT> and </TEXT> tags) of each article.

We elected to use the ACE Named Entity types PERSON, GPE (Geo-Political Entities), ORGANIZATION, VEHICLE, WEAPON, LOCATION, FACILITY, DATE, TIME, MONEY, and PERCENT. Of course, if some of these types were

not marked in an existing corpus or not needed for a given purpose, the system can easily be adapted.

Our goal was to automatically annotate the text portion of a large number of non-English articles with tags like <ENAMEX TYPE=“GPE”>Place Name</ENAMEX> as used in MUC (Message Understanding Conference). In order to do so, our system first identifies words and phrases within the text that might represent entities, primarily through the use of wikilinks. The system then uses category links and/or interwiki links to associate that phrase with an English language phrase or set of Categories. Finally, it determines the appropriate type of the English language data and assumes that the original phrase is of the same type.

In practice, the English language categorization should be treated as one-time work, since it is identical regardless of the language model being built. It is also the only stage of development at which we apply substantial linguistic knowledge, even of English.

In the sections that follow, we begin by showing how the English language categorization is done. We go on to describe how individual non-English phrases are associated with English language information. Next, we explain how possible entities are initially selected. Finally, we discuss some optional steps as well as how and why they could be used.

#### 3.2 English Language Categorization

For each article title of interest (specifically excluding Template pages, Wikipedia administrative pages, and articles whose title begins with “List of”), we extracted the categories to which that entry was assigned. Certainly, some of these category assignments are much more useful than others

For instance, we would expect that any entry in “Category:Living People” or “Category:British Lawyers” will refer to a person while any entry in “Category:Cities in Norway” will refer to a GPE. On the other hand, some are entirely unhelpful, such as “Category:1912 Establishments” which includes articles on Fenway Park (a facility), the Republic of China (a GPE), and the Better Business Bureau (an organization). Other categories can reliably be used to determine that the article does not refer to a named entity, such as “Category:Endangered species.” We manually derived a relatively small set of key phrases, the most important of which are shown in Table 1.

**Table 1: Some Useful Key Category Phrases**

PERSON	“People by”, “People in”, “People from”, “Living people”, “births”, “deaths”, “by occupation”, “Surname”, “Given names”, “Biography stub”, “human names”
ORG	“Companies”, “Teams”, “Organizations”, “Businesses”, “Media by”, “Political parties”, “Clubs”, “Advocacy groups”, “Unions”, “Corporations”, “Newspapers”, “Agencies”, “Colleges”, “Universities”, “Legislatures”, “Company stub”, “Team stub”, “University stub”, “Club stub”
GPE	“Cities”, “Countries”, “Territories”, “Counties”, “Villages”, “Municipalities”, “States” (not part of “United States”), “Republics”, “Regions”, “Settlements”
DATE	“Days”, “Months”, “Years”, “Centuries”
NONE	“Lists”, “List of”, “Wars”, “Incidents”

For each article, we searched the category hierarchy until a threshold of reliability was passed or we had reached a preset limit on how far we would search.

For example, when the system tries to classify “Jacqueline Bhabha,” it extracts the categories “British Lawyers,” “Jewish American Writers,” and “Indian Jews.” Though easily identifiable to a human, none of these matched any of our key phrases, so the system proceeded to extract the second order categories “Lawyers by nationality,” “British legal professionals,” “American writers by ethnicity,” “Jewish writers,” “Indian people by religion,” and “Indian people by ethnic or national origin” among others. “People by” is on our key phrase list, and the two occurrences passed our threshold, and she was then correctly identified.

If an article is not classified by this method, we check whether it is a disambiguation page (which often are members solely of “Category:Disambiguation”). If it is, the links within are checked to see whether there is a dominant type. For instance, the page “Amanda Foreman” is a disambiguation page, with each link on the page leading to an easily classifiable article.

Finally, we use Wiktionary, an online collaborative dictionary, to eliminate some common nouns. For example, “Tributary” is an entry in Wikipedia which would be classified as a Location if viewed solely by Category structure. However, it is found as a common noun in Wiktionary, overruling the category based result.

### 3.3 Multilingual Categorization

When attempting to categorize a non-English term that has an entry in its language’s Wikipedia, we use two techniques to make a decision based on English language information. First, whenever possible, we find the title of an associated English language article by searching for a wikilink beginning with “en:”. If such a title is found, then we categorize the English article as shown in Section 3.2, and decide that the non-English title is of the same type as its English counterpart. We note that links to/from English are the most common interlingual wikilinks.

Of course, not all articles worldwide have English equivalents (or are linked to such even if they do exist). In this case, we attempt to make a decision based on Category information, associating the categories with their English equivalents, when possible. Fortunately, many of the most useful categories have equivalents in many languages.

For example, the Breton town of Erquy has a substantial article in the French language Wikipedia, but no article in English. The system proceeds by determining that Erquy belongs to four French language categories: “Catégorie:Commune des Côtes-d’Armor,” “Catégorie:Ville portuaire de France,” “Catégorie:Port de plaisance,” and “Catégorie:Station balnéaire.” The system proceeds to associate these, respectively, with “Category:Communes of Côtes-d’Armor,” UNKNOWN, “Category:Marinas,” and “Category:Seaside resorts” by looking in the French language pages of each for wikilinks of the form [[en:...]].

The first is a subcategory of “Category:Cities, towns and villages in France” and is thus easily identified by the system as a category consisting of entities of type GPE. The other two are ambiguous categories (facility and organization elements in addition to GPE). Erquy is then determined to be a GPE by majority vote of useful categories.

We note that the second French category actually has a perfectly good English equivalent (Category:Port cities and towns in France), but no one has linked them as of this writing. We also note that the ambiguous categories are much more GPE-oriented in French. The system still makes the correct decision despite these factors.

We do not go beyond the first level categories or do any disambiguation in the non-English case. Both are avenues for future improvement.



### 3.4 The Full System

To generate a set of training data in a given language, we select a large number of articles from its Wikipedia (50,000 or more is recommended, when possible). We prepare the text by removing external links, links to images, category and interlingual links, as well as some formatting. The main processing of each article takes place in several stages, whose primary purposes are as follows:

- The first pass uses the explicit article links within the text.
- We then search an associated English language article, if available, for additional information.
- A second pass checks for multi-word phrases that exist as titles of Wikipedia articles.
- We look for certain types of person and organization instances.
- We perform additional processing for alphabetic or space-separated languages, including a third pass looking for single word Wikipedia titles.
- We use regular expressions to locate additional entities such as numeric dates.

In the first pass, we attempt to replace all wiki-links with appropriate entity tags. We assume at this stage that any phrase identified as an entity at some point in the article will be an entity of the same type throughout the article, since it is common for contributors to make the explicit link only on the first occasion that it occurs. We also assume that a phrase in a bold font within the first 100 characters is an equivalent form of the title of the article as in this start of the article on Erquy: “**Erquy (Erge-ar-Mor** en breton, **Erqi** en gallo)”. The parenthetical notation gives alternate names in the Breton and Gallo languages. (In Wiki database format, bold font is indicated by three apostrophes in succession.)

If the article has an English equivalent, we search that article for wikilinked phrases as well, on the assumption that both articles will refer to many of the same entities. As the English language Wikipedia is the largest, it frequently contains explicit references to and articles on secondary people and places mentioned, but not linked, within a given non-English article. After this point, the text to be annotated contains no Wikipedia specific information or formatting.

In the second pass, we look for strings of 2 to 4 words which were not wikilinked but which have

Wikipedia entries of their own or are partial matches to known people and organizations (i.e. “Mary Washington” in an article that contains “University of Mary Washington”). We require that each such string contains something other than a lower case letter (when a language does not use capitalization, nothing in that writing system is considered to be lower case for this purpose). When a word is in more than one such phrase, the longest match is used.

We then do some special case processing. When an organization is followed by something in parentheses such as `<ENAMEX TYPE=“ORGANIZATION”>Maktab al-Khadamāt</ENAMEX>` (MAK), we hypothesize that the text in the parentheses is an alternate name of the organization. We also looked for unmarked strings of the form X.X. followed by a capitalized word, where X represents any capital letter, and marked each occurrence as a PERSON.

For space-separated or alphabetic languages, we did some additional processing at this stage to attempt to identify more names of people. Using a list of names derived from Wiktionary (Appendix:Names) and optionally a list derived from Wikipedia (see Section 3.5.1), we mark possible parts of names. When two or more are adjacent, we mark the sequence as a PERSON. Also, we fill in partial lists of names by assuming single non-lower case words between marked names are actually parts of names themselves. That is, we would replace `<ENAMEX TYPE=“PERSON”>Fred Smith</ENAMEX>`, `Somename <ENAMEX TYPE=“PERSON”>Jones </ENAMEX>` with `<ENAMEX TYPE=“PERSON”> Fred Smith</ENAMEX>`, `<ENAMEX TYPE= “PERSON”> Somename Jones</ENAMEX>`. At this point, we performed a third pass through the article. We marked all non-lower case single words which had their own Wikipedia entry, were part of a known person's name, or were part of a known organization's name.

Afterwards, we used a series of simple, language-neutral regular expressions to find additional TIME, PERCENT, and DATE entities such as “05:30” and “12-07-05”. We also executed code that included quantities of money within a NUMEX tag, as in converting `500 <NUMEX TYPE=“MONEY”>USD</NUMEX>` into `<NUMEX TYPE=“MONEY”>500 USD</NUMEX>`.

## 3.5 Optional Processing

### 3.5.1 Recommended Additions

All of the above could be run with almost no understanding of the language being modeled (knowing whether the language was space-separated and whether it was alphabetic or character-based were the only things used). However, for most languages, we spent a small amount of time (less than one hour) browsing Wikipedia pages to improve performance in some areas.

We suggest compiling a small list of stop words. For our purposes, the determiners and the most common prepositions are sufficient, though a longer list could be used for the purpose of computational efficiency.

We also recommend compiling a list of number words as well as compiling a list of currencies, since they are not capitalized in many languages, and may not be explicitly linked either. Many languages have a page on ISO 4217 which contains all of the currency information, but the format varies sufficiently from language to language to make automatic extraction difficult. Together, these allow phrases like this (taken from the French Wikipedia) to be correctly marked in its entirety as an entity of type MONEY: “25 millions de dollars.”

If a language routinely uses honorifics such as Mr. and Mrs., that information can also be found quickly. Their use can lead to significant improvements in PERSON recognition.

During preprocessing, we typically collected a list of people names automatically, using the entity identification methods appropriate to titles of Wikipedia articles. We then used these names along with the Wiktionary derived list of names during the main processing. This does introduce some noise as the person identification is not perfect, but it ordinarily increases recall by more than it reduces precision.

### 3.5.2 Language Dependent Additions

Our usual, language-neutral processing only considers wikilinks within a single article when determining the type of unlinked words and phrases. For example, if an article included the sentence “The [[Delaware River|Delaware]] forms the boundary between [[Pennsylvania]] and [[New Jersey]]”, our system makes the assumption that every occurrence of the unlinked word “Delaware”

appearing in the same article is also referring to the river and thus mark it as a LOCATION.

For some languages, we preferred an alternate approach, best illustrated by an example: The word “Washington” without context could refer to (among others) a person, a GPE, or an organization. We could work through all of the explicit wikilinks in all articles (as a preprocessing step) whose surface form is Washington and count the number pointing to each. We could then decide that every time the word Washington appears without an explicit link, it should be marked as its most common type. This is useful for the Slavic languages, where the nominative form is typically used as the title of Wikipedia articles, while other cases appear frequently (and are rarely wikilinked).

At the same time, we can do a second type of preprocessing which allows more surface forms to be categorized. For instance, imagine that we were in a Wikipedia with no article or redirect associated to “District of Columbia” but that someone had made a wikilink of the form [[Washington|District of Columbia]]. We would then make the assumption that for all articles, District of Columbia is of the same type as Washington.

For less developed wikipeidias, this can be helpful. For languages that have reasonably well developed Wikipeidias and where entities rarely, if ever, change form for grammatical reasons (such as French), this type of preprocessing is virtually irrelevant. Worse, this processing is definitely not recommended for languages that do not use capitalization because it is not unheard of for people to include sections like: “The [[Union Station|train station]] is located at ...” which would cause the phrase “train station” to be marked as a FACILITY each time it occurred. Of course, even in languages with capitalization, “train station” would be marked incorrectly in the article in which the above was located, but the mistake would be isolated, and should have minimal impact overall.

## 4 Evaluation and Results

After each data set was generated, we used the text as a training set for input to PhoenixIDF. We had three human annotated test sets, Spanish, French and Ukrainian, consisting of newswire. When human annotated sets were not available, we held out more than 100,000 words of text generated by our wiki-mining process to use as a test set. For the above languages, we included wiki test sets for

comparison purposes. We will give our results as F-scores in the Overall, DATE, GPE, ORGANIZATION, and PERSON categories using the scoring metric in (Bikel et. al, 1999). The other ACE categories are much less common, and contribute little to the overall score.

#### 4.1 Spanish Language Evaluation

The Spanish Wikipedia is a substantial, well-developed Wikipedia, consisting of more than 290,000 articles as of October 2007. We used two test sets for comparison purposes. The first consists of 25,000 words of human annotated newswire derived from the ACE 2007 test set, manually modified to conform to our extended MUC-style standards. The second consists of 335,000 words of data generated by the Wiki process held-out during training.

**Table 2: Spanish Results**

F (prec. / recall)	Newswire	Wiki test set
<b>ALL</b>	<b>.827</b> (.851 / .805)	.846 (.843 / .848)
<b>DATE</b>	.912 (.861 / .970)	.925 (.918 / .932)
<b>GPE</b>	.877 (.914 / .843)	.877 (.886 / .868)
<b>ORG</b>	.629 (.681 / .585)	.701 (.703 / .698)
<b>PERSON</b>	.906 (.921 / .892)	.821 (.810 / .833)

There are a few particularly interesting results to note. First, because of the optional processing, recall was boosted in the PERSON category at the expense of precision. The fact that this category scores higher against newswire than against the wiki data suggests that the not-uncommon, but isolated, occurrences of non-entities being marked as PERSONs in training have little effect on the overall system. Contrarily, we note that deletions are the dominant source of error in the ORGANIZATION category, as seen by the lower recall. The better performance on the wiki set seems to suggest that either Wikipedia is relatively poor in Organizations or that PhoenixIDF underperforms when identifying Organizations relative to other categories or a combination.

An important question remains: “How do these results compare to other methodologies?” In particular, while we can get these results for free, how much work would traditional methods require to achieve comparable results?

To attempt to answer this question, we trained PhoenixIDF on additional ACE 2007 Spanish language data converted to MUC-style tags, and scored its performance using the same set of newswire. Evidently, comparable performance to our Wikipedia derived system requires between 20,000 and 40,000 words of human-annotated newswire. It is worth noting that Wikipedia itself is not newswire, so we do not have a perfect comparison.

**Table 3: Traditional Training**

~ Words of Training	Overall F-score
3500	.746
10,000	.760
20,000	<b>.807</b>
40,000	<b>.847</b>

#### 4.2 French Language Evaluation

The French Wikipedia is one of the largest Wikipedias, containing more than 570,000 articles as of October 2007. For this evaluation, we have 25,000 words of human annotated newswire (*Agence France Presse*, 30 April and 1 May 1997) covering diverse topics. We used 920,000 words of Wiki-derived data for the second test.

**Table 4: French Results**

F (prec. / recall)	Newswire	Wiki test set
<b>ALL</b>	<b>.847</b> (.877 / .819)	.844 (.847 / .840)
<b>DATE</b>	.921 (.897 / .947)	.910 (.888 / .934)
<b>GPE</b>	.907 (.933 / .882)	.868 (.889 / .849)
<b>ORG</b>	.700 (.794 / .625)	.718 (.747 / .691)
<b>PERSON</b>	.880 (.874 / .885)	.823 (.818 / .827)

The overall results seem comparable to the Spanish, with the slightly better overall performance likely correlated to the somewhat more developed Wikipedia. We did not have sufficient quantities of annotated data to run a test of the traditional methods, but Spanish and French are sufficiently similar languages that we expect this model is comparable to one created with about 40,000 words of human-annotated data.

### 4.3 Ukrainian Language Evaluation

The Ukrainian Wikipedia is a medium-sized Wikipedia with 74,000 articles as of October 2007. Also, the typical article is shorter and less well-linked to other articles than in the French or Spanish versions. Moreover, entities tend to appear in many surface forms depending on case, leading us to expect somewhat worse results. In the Ukrainian case, the newswire consisted of approximately 25,000 words from various online news sites covering primarily political topics. We also held out around 395,000 words for testing. We were also able to run a comparison test as in Spanish.

**Table 5: Ukrainian Results**

F (prec. / recall)	Newswire	Wiki test set
<b>ALL</b>	<b>.747</b> (.863 / .649)	.807 (.809 / .806)
<b>DATE</b>	.780 (.759 / .803)	.848 (.842 / .854)
<b>GPE</b>	.837 (.833 / .841)	.887 (.901 / .874)
<b>ORG</b>	.585 (.800 / .462)	.657 (.678 / .637)
<b>PERSON</b>	.764 (.899 / .664)	.690 (.675 / .706)

**Table 6: Traditional Training**

~ Words of Training	Overall F-score
5000	.662
10,000	.692
15,000	<b>.740</b>
20,000	<b>.761</b>

The Ukrainian newswire contained a much higher proportion of organizations than the French or Spanish versions, contributing to the overall lower score. The Ukrainian language Wikipedia itself contains very few articles on organizations relative to other types, so the distribution of entities of the two test sets are quite different. We also see that the Wiki-derived model performs comparably to a model trained on 15-20,000 words of human-annotated text.

### 4.4 Other Languages

For Portuguese, Russian, and Polish, we did not have human annotated corpora available for test-

ing. In each case, at least 100,000 words were held out from training to be used as a test set. It seems safe to suppose that if suitable human-annotated sets were available for testing, the PERSON score would likely be higher, and the ORGANIZATION score would likely be lower, while the DATE and GPE scores would probably be comparable.

**Table 7: Other Language Results**

F-score	Polish	Portuguese	Russian
<b>ALL</b>	<b>.859</b>	<b>.804</b>	<b>.802</b>
<b>DATE</b>	.891	.861	.822
<b>GPE</b>	.916	.826	.867
<b>ORG</b>	.785	.706	.712
<b>PERSON</b>	.836	.802	.751

## 5 Conclusions

In conclusion, we have demonstrated that Wikipedia can be used to create a Named Entity Recognition system with performance comparable to one developed from 15-40,000 words of human-annotated newswire, while not requiring any linguistic expertise on the part of the user. This level of performance, usable on its own for many purposes, can likely be obtained currently in 20-40 languages, with the expectation that more languages will become available, and that better models can be developed, as Wikipedia grows.

Moreover, it seems clear that a Wikipedia-derived system could be used as a supplement to other systems for many more languages. In particular, we have, for all practical purposes, embedded in our system an automatically generated entity dictionary.

In the future, we would like to find a way to automatically generate the list of key words and phrases for useful English language categories. This could implement the work of Kazama and Torisawa, in particular. We also believe performance could be improved by using higher order non-English categories and better disambiguation. We could also experiment with introducing automatically generated lists of entities into PhoenixIDF directly. Lists of organizations might be particularly useful, and “List of” pages are common in many languages.

## References

- Bikel, D., R. Schwartz, and R. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 211-31.
- Bunescu, R and M. Paşca. 2006. Using Encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL*, 9-16.
- Cucerzan, S. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP/CoNLL*, 708-16.
- Gabrilovitch, E. and S. Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of IJCAI*, 1606-11.
- Gabrilovitch, E. and S. Markovitch. 2006. Overcoming the brittleness bottleneck using Wikipedia: enhancing text categorization with encyclopedic knowledge. In *Proceedings of AAAI*, 1301-06.
- Gabrilovitch, E. and S. Markovitch. 2005. Feature generation for text categorization using world knowledge. In *Proceedings of IJCAI*, 1048-53.
- Kazama, J. and K. Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of EMNLP/CoNLL*, 698-707.
- Milne, D., O. Medelyan and I. Witten. 2006. Mining domain-specific thesauri from Wikipedia: a case study. *Web Intelligence 2006*, 442-48
- Strube, M. and S. P. Ponzeto. 2006. WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of AAAI*, 1419-24.
- Toral, A. and R. Muñoz. 2006. A proposal to automatically build and maintain gazetteers for named entity recognition by using Wikipedia. In *Proceedings of EACL*, 56-61.
- Weale, T. 2006. Using Wikipedia categories for document classification. *Ohio St. University, preprint*.
- Zesch, T., I. Gurevych and M. Mühlhäuser. 2007. Analyzing and accessing Wikipedia as a lexical semantic resource. In *Proceedings of GLDV*, 213-21.

# Distributional Identification of Non-Referential Pronouns

**Shane Bergsma**

Department of Computing Science  
University of Alberta  
Edmonton, Alberta  
Canada, T6G 2E8  
bergsma@cs.ualberta.ca

**Dekang Lin**

Google, Inc.  
1600 Amphitheatre Parkway  
Mountain View  
California, 94301  
lindek@google.com

**Randy Goebel**

Department of Computing Science  
University of Alberta  
Edmonton, Alberta  
Canada, T6G 2E8  
goebel@cs.ualberta.ca

## Abstract

We present an automatic approach to determining whether a pronoun in text refers to a preceding noun phrase or is instead *non-referential*. We extract the surrounding textual context of the pronoun and gather, from a large corpus, the distribution of words that occur within that context. We learn to reliably classify these distributions as representing either referential or non-referential pronoun instances. Despite its simplicity, experimental results on classifying the English pronoun *it* show the system achieves the highest performance yet attained on this important task.

## 1 Introduction

The goal of coreference resolution is to determine which noun phrases in a document refer to the same real-world entity. As part of this task, coreference resolution systems must decide which pronouns refer to preceding noun phrases (called antecedents) and which do not. In particular, a long-standing challenge has been to correctly classify instances of the English pronoun *it*. Consider the sentences:

- (1) You can make it in advance.
- (2) You can make it in Hollywood.

In sentence (1), *it* is an anaphoric pronoun referring to some previous noun phrase, like “the sauce” or “an appointment.” In sentence (2), *it* is part of the idiomatic expression “make it” meaning “succeed.” A coreference resolution system should find an antecedent for the first *it* but not the second. Pronouns

that do not refer to preceding noun phrases are called *non-anaphoric* or *non-referential* pronouns.

The word *it* is one of the most frequent words in the English language, accounting for about 1% of tokens in text and over a quarter of all third-person pronouns.<sup>1</sup> Usually between a quarter and a half of *it* instances are non-referential (e.g. Section 4, Table 3). As with other pronouns, the preceding discourse can affect *it*’s interpretation. For example, sentence (2) can be interpreted as referential if the preceding sentence is “You want to make a movie?” We show, however, that we can reliably classify a pronoun as being referential or non-referential based solely on the local context surrounding the pronoun.

We do this by turning the context into patterns and enumerating all the words that can take the place of *it* in these patterns. For sentence (1), we can extract the context pattern “make \* in advance” and for sentence (2) “make \* in Hollywood,” where “\*” is a wildcard that can be filled by any token. Non-referential distributions tend to have the word *it* filling the wildcard position. Referential distributions occur with many other noun phrase fillers. For example, in our n-gram collection (Section 3.4), “make it in advance” and “make them in advance” occur roughly the same number of times (442 vs. 449), indicating a referential pattern. In contrast, “make it in Hollywood” occurs 3421 times while “make them in Hollywood” does not occur at all.

These simple counts strongly indicate whether another noun can replace the pronoun. Thus we can computationally distinguish between a) pronouns that refer to nouns, and b) all other instances: including those that have no antecedent, like sentence (2),

<sup>1</sup>e.g. <http://ucrel.lancs.ac.uk/bncfreq/flists.html>

and those that refer to sentences, clauses, or implied topics of discourse. Beyond the practical value of this distinction, Section 3 provides some theoretical justification for our binary classification.

Section 3 also shows how to automatically extract and collect counts for context patterns, and how to combine the information using a machine learned classifier. Section 4 describes our data for learning and evaluation, *It-Bank*: a set of over three thousand labelled instances of the pronoun *it* from a variety of text sources. Section 4 also explains our comparison approaches and experimental methodology. Section 5 presents our results, including an interesting comparison of our system to human classification given equivalent segments of context.

## 2 Related Work

The difficulty of non-referential pronouns has been acknowledged since the beginning of computational resolution of anaphora. Hobbs (1978) notes his algorithm does not handle pronominal references to sentences nor cases where *it* occurs in time or weather expressions. Hirst (1981, page 17) emphasizes the importance of detecting non-referential pronouns, “lest precious hours be lost in bootless searches for textual referents.” Müller (2006) summarizes the evolution of computational approaches to non-referential *it* detection. In particular, note the pioneering work of Paice and Husk (1987), the inclusion of non-referential *it* detection in a full anaphora resolution system by Lappin and Leass (1994), and the machine learning approach of Evans (2001).

There has recently been renewed interest in non-referential pronouns, driven by three primary sources. First of all, research in coreference resolution has shown the benefits of modules for general noun anaphoricity determination (Ng and Cardie, 2002; Denis and Baldridge, 2007). Unfortunately, these studies handle pronouns inadequately; judging from the decision trees and performance figures, Ng and Cardie (2002)’s system treats all pronouns as anaphoric by default. Secondly, while most pronoun resolution evaluations simply exclude non-referential pronouns, recent unsupervised approaches (Cherry and Bergsma, 2005; Haghighi and Klein, 2007) must deal with all pronouns in unrestricted text, and therefore need robust modules to

automatically handle non-referential instances. Finally, reference resolution has moved beyond written text into spoken dialog. Here, non-referential pronouns are pervasive. Eckert and Strube (2000) report that in the Switchboard corpus, only 45% of demonstratives and third-person pronouns have a noun phrase antecedent. Handling the common non-referential instances is thus especially vital.

One issue with systems for non-referential detection is the amount of language-specific knowledge that must be encoded. Consider a system that jointly performs anaphora resolution and word alignment in parallel corpora for machine translation. For this task, we need to identify non-referential anaphora in multiple languages. It is not always clear to what extent the features and modules developed for English systems apply to other languages. For example, the detector of Lappin and Leass (1994) labels a pronoun as non-referential if it matches one of several syntactic patterns, including: “It is **Cogv-ed** that **Sentence**,” where **Cogv** is a “cognitive verb” such as *recommend*, *think*, *believe*, *know*, *anticipate*, etc. Porting this approach to a new language would require not only access to a syntactic parser and a list of cognitive verbs in that language, but the development of new patterns to catch non-referential pronoun uses that do not exist in English.

Moreover, writing a set of rules to capture this phenomenon is likely to miss many less-common uses. Alternatively, recent machine-learning approaches leverage a more general representation of a pronoun instance. For example, Müller (2006) has a feature for “distance to next complementizer (*that*, *if*, *whether*)” and features for the tokens and part-of-speech tags of the context words. Unfortunately, there is still a lot of implicit and explicit English-specific knowledge needed to develop these features, including, for example, lists of “seem” verbs such as *appear*, *look*, *mean*, *happen*. Similarly, the machine-learned system of Boyd et al. (2005) uses a set of “idiom patterns” like “*on the face of it*” that trigger binary features if detected in the pronoun context. Although machine learned systems can flexibly balance the various indicators and contra-indicators of non-referentiality, a particular feature is only useful if it is relevant to an example in limited labelled training data.

Our approach avoids hand-crafting a set of spe-

cific indicator features; we simply use the distribution of the pronoun’s context. Our method is thus related to previous work based on Harris (1985)’s distributional hypothesis.<sup>2</sup> It has been used to determine both word and syntactic path similarity (Hindle, 1990; Lin, 1998a; Lin and Pantel, 2001). Our work is part of a trend of extracting other important information from statistical distributions. Dagan and Itai (1990) use the distribution of a pronoun’s context to determine which candidate antecedents can fit the context. Bergsma and Lin (2006) determine the likelihood of coreference along the syntactic path connecting a pronoun to a possible antecedent, by looking at the distribution of the path in text. These approaches, like ours, are ways to inject sophisticated “world knowledge” into anaphora resolution.

### 3 Methodology

#### 3.1 Definition

Our approach distinguishes contexts where pronouns cannot be replaced by a preceding noun phrase (non-noun-referential) from those where nouns can occur (noun-referential). Although coreference evaluations, such as the MUC (1997) tasks, also make this distinction, it is not necessarily used by all researchers. Evans (2001), for example, distinguishes between “clause anaphoric” and “pleonastic” as in the following two instances:

- (3) The paper reported that it had snowed. *It* was obvious. (*clause anaphoric*)
- (4) *It* was obvious that it had snowed. (*pleonastic*)

The word *It* in sentence (3) is considered referential, while the word *It* in sentence (4) is considered non-referential.<sup>3</sup> From our perspective, this interpretation is somewhat arbitrary. One could also say that the *It* in both cases refers to the clause “that it had snowed.” Indeed, annotation experiments using very fine-grained categories show low annotation reliability (Müller, 2006). On the other hand, there is no debate over the importance nor the definition of distinguishing pronouns that refer to nouns from those that do not. We adopt this distinction for our

work, and show it has good inter-annotator reliability (Section 4.1). We henceforth refer to non-noun-referential simply as non-referential, and thus consider the word *It* in both sentences (3) and (4) as non-referential.

Non-referential pronouns are widespread in natural language. The *es* in the German “Wie geht es Ihnen” and the *il* in the French “S’il vous plaît” are both non-referential. In pro-drop languages that may omit subject pronouns, there remains the question of whether an omitted pronoun is referential (Zhao and Ng, 2007). Although we focus on the English pronoun *it*, our approach should differentiate any words that have both a structural and a referential role in language, e.g. words like *this*, *there* and *that* (Müller, 2007). We believe a distributional approach could also help in related tasks like identifying the generic use of *you* (Gupta et al., 2007).

#### 3.2 Context Distribution

Our method extracts the context surrounding a pronoun and determines which other words can take the place of the pronoun in the context. The extracted segments of context are called *context patterns*. The words that take the place of the pronoun are called *pattern fillers*. We gather pattern fillers from a large collection of n-gram frequencies. The maximum size of a context pattern depends on the size of n-grams available in the data. In our n-gram collection (Section 3.4), the lengths of the n-grams range from unigrams to 5-grams, so our maximum pattern size is five. For a particular pronoun in text, there are five possible 5-grams that span the pronoun. For example, in the following instance of *it*:

... said here Thursday that it is unnecessary to continue ...  
We can extract the following 5-gram patterns:

```
said here Thursday that *
here Thursday that * is
Thursday that * is unnecessary
that * is unnecessary to
* is unnecessary to continue
```

Similarly, we extract the four 4-gram patterns. Shorter n-grams were not found to improve performance on development data and hence are not extracted. We only use context within the current sentence (including the beginning-of-sentence and end-of-sentence tokens) so if a pronoun occurs near a sentence boundary, some patterns may be missing.

<sup>2</sup>Words occurring in similar contexts have similar meanings

<sup>3</sup>The *it* in “it had snowed” is, of course, non-referential.



Pattern Filler Type	String
#1: 3rd-person pron. sing.	<i>it/its</i>
#2: 3rd-person pron. plur.	<i>they/them/their</i>
#3: any other pronoun	<i>he/him/his/, I/me/my, etc.</i>
#4: infrequent word token	$\langle UNK \rangle$
#5: any other token	*

Table 1: Pattern filler types

We take a few steps to improve generality. We change the patterns to lower-case, convert sequences of digits to the # symbol, and run the Porter stemmer<sup>4</sup> (Porter, 1980). To generalize rare names, we convert capitalized words longer than five characters to a special *NE* tag. We also added a few simple rules to stem the irregular verbs *be*, *have*, *do*, and *said*, and convert the common contractions *'nt*, *'s*, *'m*, *'re*, *'ve*, *'d*, and *'ll* to their most likely stem.

We do the same processing to our n-gram corpus. We then find all n-grams matching our patterns, allowing any token to match the wildcard in place of *it*. Also, other pronouns in the pattern are allowed to match a corresponding pronoun in an n-gram, regardless of differences in inflection and class.

We now discuss how to use the distribution of pattern fillers. For identifying non-referential *it* in English, we are interested in how often *it* occurs as a pattern filler versus other *nouns*. However, determining part-of-speech in a large n-gram corpus is not simple, nor would it easily extend to other languages. Instead, we gather counts for five different classes of words that fill the wildcard position, easily determined by string match (Table 1). The third-person plural *they* (#2) reliably occurs in patterns where referential *it* also resides. The occurrence of *any other pronoun* (#3) guarantees that at the very least the pattern filler is a noun. A match with the infrequent word token  $\langle UNK \rangle$  (#4) (explained in Section 3.4) will likely be a noun because nouns account for a large proportion of rare words in a corpus. Gathering *any other token* (#5) also mostly finds nouns; inserting another part-of-speech usually

<sup>4</sup>Adapted from the Bow-toolkit (McCallum, 1996). Our method also works without the stemmer; we simply truncate the words in the pattern at a given maximum length (see Section 5.1). With simple truncation, all the pattern processing can be easily applied to other languages.

Pattern	Filler Counts			
	#1	#2	#3	#5
sai here <i>NE</i> that *	84	0	291	3985
here <i>NE</i> that * be	0	0	0	93
<i>NE</i> that * be unnecessari	0	0	0	0
that * be unnecessari to	16726	56	0	228
* be unnecessari to continu	258	0	0	0

Table 2: 5-gram context patterns and pattern-filler counts for the Section 3.2 example.

results in an unlikely, ungrammatical pattern.

Table 2 gives the stemmed context patterns for our running example. It also gives the n-gram counts of pattern fillers matching the first four filler types (there were no matches of the  $\langle UNK \rangle$  type, #4).

### 3.3 Feature Vector Representation

There are many possible ways to use the above counts. Intuitively, our method should identify as non-referential those instances that have a high proportion of fillers of type #1 (i.e., the word *it*), while labelling as referential those with high counts for other types of fillers. We would also like to leverage the possibility that some of the patterns may be more predictive than others, depending on where the wildcard lies in the pattern. For example, in Table 2, the cases where the *it*-position is near the beginning of the pattern best reflect the non-referential nature of this instance. We can achieve these aims by ordering the counts in a feature vector, and using a labelled set of training examples to learn a classifier that optimally weights the counts.

For classification, we define non-referential as positive and referential as negative. Our feature representation very much resembles Table 2. For each of the five 5-gram patterns, ordered by the position of the wildcard, we have features for the logarithm of counts for filler types #1, #2, ... #5. Similarly, for each of the four 4-gram patterns, we provide the log-counts corresponding to types #1, #2, ... #5 as well. Before taking the logarithm, we smooth the counts by adding a fixed number to all observed values. We also provide, for each pattern, a feature that indicates if the pattern is not available because the *it*-position would cause the pattern to span beyond the current sentence. There are twenty-five 5-gram, twenty 4-gram, and nine indicator features in total.

Our classifier should learn positive weights on the type #1 counts and negative weights on the other types, with higher absolute weights on the more predictive filler types and pattern positions. Note that leaving the pattern counts unnormalized automatically allows patterns with higher counts to contribute more to the prediction of their associated instances.

### 3.4 N-Gram Data

We now describe the collection of n-grams and their counts used in our implementation. We use, to our knowledge, the largest publicly available collection: the Google Web 1T 5-gram Corpus Version 1.1.<sup>5</sup> This collection was generated from approximately 1 trillion tokens of online text. In this data, tokens appearing less than 200 times have been mapped to the  $\langle \text{UNK} \rangle$  symbol. Also, only n-grams appearing more than 40 times are included. For languages where such an extensive n-gram resource is not available, the n-gram counts could also be taken from the page-counts returned by an Internet search engine.

## 4 Evaluation

### 4.1 Labelled *It* Data

We need labelled data for training and evaluation of our system. This data indicates, for every occurrence of the pronoun *it*, whether it refers to a preceding noun phrase or not. Standard coreference resolution data sets annotate all noun phrases that have an antecedent noun phrase in the text. Therefore, we can extract labelled instances of *it* from these sets. We do this for the dry-run and formal sets from MUC-7 (1997), and merge them into a single data set.

Of course, full coreference-annotated data is a precious resource, with the pronoun *it* making up only a small portion of the marked-up noun phrases. We thus created annotated data specifically for the pronoun *it*. We annotated 1020 instances in a collection of Science News articles (from 1995-2000), downloaded from the Science News website. We also annotated 709 instances in the WSJ portion of the DARPA TIPSTER Project (Harman, 1992), and 279 instances in the English portion of the Europarl Corpus (Koehn, 2005).

A single annotator ( $A_1$ ) labelled all three data sets, while two additional annotators not connected

Data Set	Number of <i>It</i>	% Non-Referential
<i>Europarl</i>	279	50.9
<i>Sci-News</i>	1020	32.6
<i>WSJ</i>	709	25.1
<i>MUC</i>	129	31.8
<i>Train</i>	1069	33.2
<i>Test</i>	1067	31.7
<i>Test-200</i>	200	30.0

Table 3: Data sets used in experiments.

with the project ( $A_2$  and  $A_3$ ) were asked to separately re-annotate a portion of each, so that inter-annotator agreement could be calculated.  $A_1$  and  $A_2$  agreed on 96% of annotation decisions, while  $A_1$ - $A_3$ , and  $A_2$ - $A_3$ , agreed on 91% and 93% of decisions, respectively. The *Kappa* statistic (Jurafsky and Martin, 2000, page 315), with  $P(E)$  computed from the confusion matrices, was a high 0.90 for  $A_1$ - $A_2$ , and 0.79 and 0.81 for the other pairs, around the 0.80 considered to be good reliability. These are, perhaps surprisingly, the only known *it*-annotation-agreement statistics available for written text. They contrast favourably with the low agreement seen on categorizing *it* in spoken dialog (Müller, 2006).

We make all the annotations available in *It-Bank*, an online repository for annotated *it*-instances.<sup>6</sup> *It-Bank* also allows other researchers to distribute their *it* annotations. Often, the full text of articles containing annotations cannot be shared because of copyright. However, sharing just the sentences containing the word *it*, randomly-ordered, is permissible under fair-use guidelines. The original annotators retain their copyright on the annotations.

We use our annotated data in two ways. First of all, we perform cross-validation experiments on each of the data sets individually, to help gauge the difficulty of resolution on particular domains and volumes of training data. Secondly, we randomly distribute all instances into two main sets, a training set and a test set. We also construct a smaller test set, *Test-200*, containing only the first 200 instances in the *Test* set. We use *Test-200* for human experiments and error analysis (Section 5.2). Table 3 summarizes all the sets used in the experiments.

<sup>5</sup>Available from the LDC as LDC2006T13

<sup>6</sup>[www.cs.ualberta.ca/~bergma/ItBank/](http://www.cs.ualberta.ca/~bergma/ItBank/). *It-Bank* also contains an additional 1,077 examples used as development data.

## 4.2 Comparison Approaches

We represent feature vectors exactly as described in Section 3.3. We smooth by adding 40 to all counts, equal to the minimum count in the n-gram data. For classification, we use a maximum entropy model (Berger et al., 1996), from the logistic regression package in Weka (Witten and Frank, 2005), with all default parameter settings. Results with our distributional approach are labelled as DISTRIB. Note that our maximum entropy classifier actually produces a *probability* of non-referentiality, which is thresholded at 50% to make a classification.

As a baseline, we implemented the non-referential *it* detector of Lappin and Leass (1994), labelled as LL in the results. This is a *syntactic* detector, a point missed by Evans (2001) in his criticism: the patterns are robust to intervening words and modifiers (e.g. “it was *never* thought *by the committee* that...”) provided the sentence is parsed correctly.<sup>7</sup> We automatically parse sentences with Minipar, a broad-coverage dependency parser (Lin, 1998b).

We also use a separate, extended version of the LL detector, implemented for large-scale non-referential detection by Cherry and Bergsma (2005). This system, also for Minipar, additionally detects instances of *it* labelled with Minipar’s pleonastic category *Subj*. It uses Minipar’s named-entity recognition to identify time expressions, such as “it was midnight,” and provides a number of other patterns to match common non-referential *it* uses, such as in expressions like “darn it,” “don’t overdo it,” etc. This extended detector is labelled as MINIPL (for Minipar pleonasticity) in our results.

Finally, we tested a system that combines the above three approaches. We simply add the LL and MINIPL decisions as binary features in the DISTRIB system. This system is called COMBO in our results.

## 4.3 Evaluation Criteria

We follow Müller (2006)’s evaluation criteria. Precision (P) is the proportion of instances that we label as non-referential that are indeed non-referential. Recall (R) is the proportion of true non-referentials that we detect, and is thus a measure of the coverage

<sup>7</sup>Our approach, on the other hand, would seem to be susceptible to such intervening material, if it pushes indicative context tokens out of the 5-token window.

System	P	R	F	Acc
LL	<b>93.4</b>	21.0	34.3	74.5
MINIPL	66.4	49.7	56.9	76.1
DISTRIB	81.4	71.0	75.8	85.7
COMBO	81.3	<b>73.4</b>	<b>77.1</b>	<b>86.2</b>

Table 4: *Train/Test*-split performance (%).

of the system. F-Score (F) is the geometric average of precision and recall; it is the most common non-referential detection metric. Accuracy (Acc) is the percentage of instances labelled correctly.

## 5 Results

### 5.1 System Comparison

Table 4 gives precision, recall, F-score, and accuracy on the *Train/Test* split. Note that while the LL system has high detection precision, it has very low recall, sharply reducing F-score. The MINIPL approach sacrifices some precision for much higher recall, but again has fairly low F-score. To our knowledge, our COMBO system, with an F-Score of 77.1%, achieves the highest performance of any non-referential system yet implemented. Even more importantly, DISTRIB, which requires only minimal linguistic processing and no encoding of specific indicator patterns, achieves 75.8% F-Score. The difference between COMBO and DISTRIB is not statistically significant, while both are significantly better than the rule-based approaches.<sup>8</sup> This provides strong motivation for a “light-weight” approach to non-referential *it* detection – one that does not require parsing or hand-crafted rules and – is easily ported to new languages and text domains.

Since applying an English stemmer to the context words (Section 3.2) reduces the portability of the distributional technique, we investigated the use of more portable pattern abstraction. Figure 1 compares the use of the stemmer to simply truncating the words in the patterns at a certain maximum length. Using no truncation (Unaltered) drops the F-Score by 4.3%, while truncating the patterns to a length of four only drops the F-Score by 1.4%, a difference which is not statistically significant. Simple truncation may be a good option for other languages where stemmers are not readily available. The optimum

<sup>8</sup>All significance testing uses McNemar’s test,  $p < 0.05$

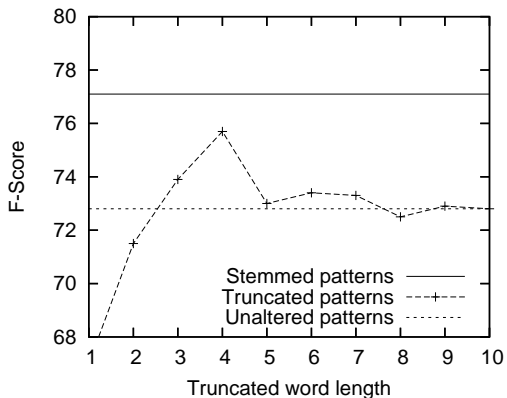


Figure 1: Effect of pattern-word truncation on non-referential *it* detection (COMBO system, *Train/Test* split).

System	<i>Europl.</i>	<i>Sci-News</i>	<i>WSJ</i>	<i>MUC</i>
LL	44.0	39.3	21.5	13.3
MINIPL	70.3	61.8	22.0	50.7
DISTRIB	79.7	77.2	<b>69.5</b>	68.2
COMBO	76.2	<b>78.7</b>	68.1	65.9
COMBO4	<b>83.6</b>	76.5	67.1	<b>74.7</b>

Table 5: 10-fold cross validation F-Score (%).

truncation size will likely depend on the length of the base forms of words in that language. For real-world application of our approach, truncation also reduces the table sizes (and thus storage and lookup costs) of any pre-compiled *it*-pattern database.

Table 5 compares the 10-fold cross-validation F-score of our systems on the four data sets. The performance of COMBO on *Europarl* and *MUC* is affected by the small number of instances in these sets (Section 4, Table 3). We can reduce data fragmentation by removing features. For example, if we only use the length-4 patterns in COMBO (labelled as COMBO4), performance increases dramatically on *Europarl* and *MUC*, while dipping slightly for the larger *Sci-News* and *WSJ* sets. Furthermore, selecting just the three most useful filler type counts as features (#1,#2,#5), boosts F-Score on *Europarl* to 86.5%, 10% above the full COMBO system.

## 5.2 Analysis and Discussion

In light of these strong results, it is worth considering where further gains in performance might yet be found. One key question is to what extent a limited context restricts identification performance. We first tested the importance of the pattern length by

System	P	R	F	Acc
DISTRIB	80.0	73.3	76.5	86.5
COMBO	80.7	76.7	78.6	87.5
Human-1	92.7	63.3	75.2	87.5
Human-2	84.0	70.0	76.4	87.0
Human-3	72.2	86.7	78.8	86.0

Table 6: Evaluation on *Test-200* (%).

using only the length-4 counts in the DISTRIB system (*Train/Test* split). Surprisingly, the drop in F-Score was only one percent, to 74.8%. Using only the length-5 counts drops F-Score to 71.4%. Neither are statistically significant; however there seems to be diminishing returns from longer context patterns.

Another way to view the limited context is to ask, given the amount of context we have, are we making optimum use of it? We answer this by seeing how well humans can do with the same information. As explained in Section 3.2, our system uses 5-gram context patterns that together span from four-to-the-left to four-to-the-right of the pronoun. We thus provide these same nine-token windows to our human subjects, and ask them to decide whether the pronouns refer to previous noun phrases or not, based on these contexts. Subjects first performed a dry-run experiment on separate development data. They were shown their errors and sources of confusion were clarified. They then made the judgments unassisted on the final *Test-200* data. Three humans performed the experiment. Their results show a range of preferences for precision versus recall, with both F-Score and Accuracy on average below the performance of COMBO (Table 6). Foremost, these results show that our distributional approach is already getting good leverage from the limited context information, around that achieved by our best human.

It is instructive to inspect the twenty-five *Test-200* instances that the COMBO system classified incorrectly, given human performance on this same set. Seventeen of the twenty-five COMBO errors were also made by one or more human subjects, suggesting system errors are also mostly due to limited context. For example, one of these errors was for the context: “it takes an astounding amount...” Here, the non-referential nature of the instance is not apparent without the infinitive clause that ends the sentence: “... of time to compare very long DNA sequences

with each other.”

Six of the eight errors unique to the COMBO system were cases where the system falsely said the pronoun was non-referential. Four of these could have referred to entire sentences or clauses rather than nouns. These confusing cases, for both humans and our system, result from our definition of a referential pronoun: pronouns with verbal or clause antecedents are considered non-referential (Section 3.1). If an antecedent verb or clause is replaced by a nominalization (*Smith researched...* to *Smith’s research*), a referring pronoun, in the same context, becomes referential. When we inspect the probabilities produced by the maximum entropy classifier (Section 4.2), we see only a weak bias for the non-referential class on these examples, reflecting our classifier’s uncertainty. It would likely be possible to improve accuracy on these cases by encoding the presence or absence of preceding nominalizations as a feature of our classifier.

Another false non-referential decision is for the phrase “... machine he had installed it on.” The *it* is actually referential, but the extracted patterns (e.g. “he had install \* on”) are nevertheless usually filled with *it*.<sup>9</sup> Again, it might be possible to fix such examples by leveraging the preceding discourse. Notably, the first noun-phrase before the context is the word “software.” There is strong compatibility between the pronoun-parent “install” and the candidate antecedent “software.” In a full coreference resolution system, when the anaphora resolution module has a strong preference to link *it* to an antecedent (which it should when the pronoun is indeed referential), we can override a weak non-referential probability. Non-referential *it* detection should not be a pre-processing step, but rather part of a globally-optimal configuration, as was done for general noun phrase anaphoricity by Denis and Baldridge (2007).

The suitability of this kind of approach to correcting some of our system’s errors is especially obvious when we inspect the probabilities of the maximum entropy model’s output decisions on the *Test-200* set. Where the maximum entropy classifier makes mistakes, it does so with less confidence than when it classifies correct examples. The average predicted

---

<sup>9</sup>This example also suggests using filler counts for the word “the” as a feature when *it* is the last word in the pattern.

probability of the incorrect classifications is 76.0% while the average probability of the correct classifications is 90.3%. Many incorrect decisions are ready to switch sides; our next step will be to use features of the preceding discourse and the candidate antecedents to help give them a push.

## 6 Conclusion

We have presented an approach to detecting non-referential pronouns in text based on the distribution of the pronoun’s context. The approach is simple to implement, attains state-of-the-art results, and should be easily ported to other languages. Our technique demonstrates how large volumes of data can be used to gather world knowledge for natural language processing. A consequence of this research was the creation of *It-Bank*, a collection of thousands of labelled examples of the pronoun *it*, which will benefit other coreference resolution researchers.

Error analysis reveals that our system is getting good leverage out of the pronoun context, achieving results comparable to human performance given equivalent information. To boost performance further, we will need to incorporate information from preceding discourse. Future research will also test the distributional classification of other ambiguous pronouns, like *this*, *you*, *there*, and *that*. Another avenue of study will look at the interaction between coreference resolution and machine translation. For example, if a single form in English (e.g. *that*) is separated into different meanings in another language (e.g., Spanish demonstrative *ese*, nominal reference *ése*, abstract or statement reference *eso*, and complementizer *que*), then aligned examples provide automatically-disambiguated English data. We could extract context patterns and collect statistics from these examples like in our current approach. In general, jointly optimizing translation and coreference is an exciting and largely unexplored research area, now partly enabled by our portable non-referential detection methodology.

## Acknowledgments

We thank Kristin Musselman and Christopher Pinchak for assistance preparing the data, and we thank Google Inc. for sharing their 5-gram corpus. We gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada, the Alberta Ingenuity Fund, and the Alberta Informatics Circle of Research Excellence.

## References

- Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.
- Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *COLING-ACL*, pages 33–40.
- Adrienne Boyd, Whitney Gegg-Harrison, and Donna Byron. 2005. Identifying non-referential *it*: a machine learning approach incorporating linguistically motivated patterns. In *ACL Workshop on Feature Engineering for Machine Learning in NLP*, pages 40–47.
- Colin Cherry and Shane Bergsma. 2005. An expectation maximization approach to pronoun resolution. In *CoNLL*, pages 88–95.
- Ido Dagan and Alan Itai. 1990. Automatic processing of large corpora for the resolution of anaphora references. In *COLING*, volume 3, pages 330–332.
- Pascal Denis and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference using integer programming. In *NAACL-HLT*, pages 236–243.
- Miriam Eckert and Michael Strube. 2000. Dialogue acts, synchronizing units, and anaphora resolution. *Journal of Semantics*, 17(1):51–89.
- Richard Evans. 2001. Applying machine learning toward an automatic classification of *it*. *Literary and Linguistic Computing*, 16(1):45–57.
- Surabhi Gupta, Matthew Purver, and Dan Jurafsky. 2007. Disambiguating between generic and referential “you” in dialog. In *ACL Demo and Poster Sessions*, pages 105–108.
- Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric Bayesian model. In *ACL*, pages 848–855.
- Donna Harman. 1992. The DARPA TIPSTER project. *ACM SIGIR Forum*, 26(2):26–28.
- Zellig Harris. 1985. Distributional structure. In J.J. Katz, editor, *The Philosophy of Linguistics*, pages 26–47. Oxford University Press, New York.
- Donald Hindle. 1990. Noun classification from predicate-argument structures. In *ACL*, pages 268–275.
- Graeme Hirst. 1981. *Anaphora in Natural Language Understanding: A Survey*. Springer Verlag.
- Jerry Hobbs. 1978. Resolving pronoun references. *Lingua*, 44(311):339–352.
- Daniel Jurafsky and James H. Martin. 2000. *Speech and language processing*. Prentice Hall.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit X*, pages 79–86.
- Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.
- Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.
- Dekang Lin. 1998a. Automatic retrieval and clustering of similar words. In *COLING-ACL*, pages 768–773.
- Dekang Lin. 1998b. Dependency-based evaluation of MINIPAR. In *LREC Workshop on the Evaluation of Parsing Systems*.
- Andrew Kachites McCallum. 1996. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow>.
- MUC-7. 1997. Coreference task definition (v3.0, 13 Jul 97). In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.
- Christoph Müller. 2006. Automatic detection of non-referential *It* in spoken multi-party dialog. In *EACL*, pages 49–56.
- Christoph Müller. 2007. Resolving *It*, *This*, and *That* in unrestricted multi-party dialog. In *ACL*, pages 816–823.
- Vincent Ng and Claire Cardie. 2002. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *COLING*, pages 730–736.
- Chris D. Paice and Gareth D. Husk. 1987. Towards the automatic recognition of anaphoric features in English text: the impersonal pronoun “it”. *Computer Speech and Language*, 2:109–132.
- Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.
- Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, second edition.
- Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *EMNLP*, pages 541–550.

# Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs

**Marius Paşca**

Google Inc.  
Mountain View, California 94043  
mars@google.com

**Benjamin Van Durme\***

University of Rochester  
Rochester, New York 14627  
vandurme@cs.rochester.edu

## Abstract

A new approach to large-scale information extraction exploits both Web documents and query logs to acquire thousands of open-domain classes of instances, along with relevant sets of open-domain class attributes at precision levels previously obtained only on small-scale, manually-assembled classes.

## 1 Introduction

Current methods for large-scale information extraction take advantage of unstructured text available from either Web documents (Banko et al., 2007; Snow et al., 2006) or, more recently, logs of Web search queries (Paşca, 2007) to acquire useful knowledge with minimal supervision. Given a manually-specified target attribute (e.g., birth years for people) and starting from as few as 10 seed facts such as (e.g., *John Lennon, 1941*), as many as a million facts of the same type can be derived from unstructured text within Web documents (Paşca et al., 2006). Similarly, given a manually-specified target class (e.g., *Drug*) with its instances (e.g., *Vicodin* and *Xanax*) and starting from as few as 5 seed attributes (e.g., *side effects* and *maximum dose* for *Drug*), other relevant attributes can be extracted for the same class from query logs (Paşca, 2007). These and other previous methods require the manual specification of the input classes of instances before any knowledge (e.g., facts or attributes) can be acquired for those classes.

The extraction method introduced in this paper mines a collection of Web search queries and a collection of Web documents to acquire open-domain classes in the form of instance sets (e.g., {*whales, seals, dolphins, sea lions,...*}) associated with class labels (e.g., *marine animals*), as well as large sets of open-domain attributes for each class (e.g., *circulatory system, life cycle, evolution, food chain* and *scientific name* for the class *marine animals*). In this light, the contributions of this paper are four-fold. First, instead of separately addressing the tasks of collecting unlabeled sets of instances (Lin, 1998), assigning appropriate class labels to a given set of instances (Pantel and Ravichandran, 2004), and identifying relevant attributes for a given set of classes (Paşca, 2007), our integrated method from Section 2 enables the *simultaneous extraction* of class instances, associated labels and attributes. Second, by exploiting the contents of query logs during the extraction of labeled classes of instances from Web documents, we acquire thousands (4,583, to be exact) of *open-domain classes* covering a wide range of topics and domains. The accuracy reported in Section 3.2 exceeds 80% for both instance sets and class labels, although the extraction of classes requires a remarkably small amount of supervision, in the form of only a few commonly-used Is-A extraction patterns. Third, we conduct the first study in extracting attributes for thousands of open-domain, *automatically-acquired* classes, at precision levels over 70% at rank 10, and 67% at rank 20 as described in Section 3.3. The amount of supervision is limited to five seed attributes provided for only one reference class. In comparison, the largest previous

---

\*Contributions made during an internship at Google.

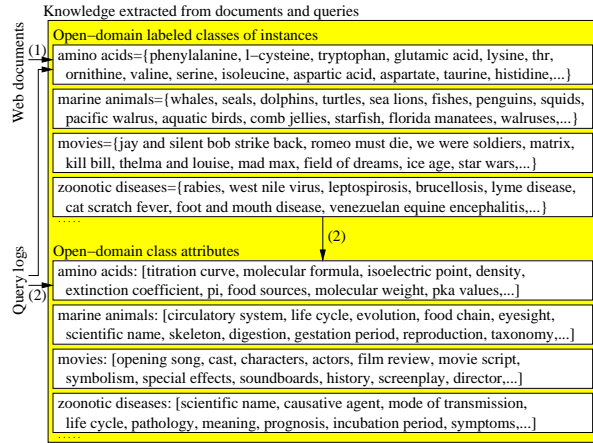


Figure 1: Overview of weakly-supervised extraction of class instances, class labels and class attributes from Web documents and query logs

study in attribute extraction reports results on a set of 40 *manually-assembled* classes, and requires five seed attributes to be provided as input for each class. Fourth, we introduce the first approach to information extraction from a combination of *both* Web documents and search query logs, to extract open-domain knowledge that is expected to be suitable for later use. In contrast, the textual data sources used in previous studies in large-scale information extraction are *either* Web documents (Mooney and Bunescu, 2005; Banko et al., 2007) *or*, recently, query logs (Paşca, 2007), but not both.

## 2 Extraction from Documents and Queries

### 2.1 Open-Domain Labeled Classes of Instances

Figure 1 provides an overview of how Web documents and queries are used together to acquire open-domain, labeled classes of instances (phase (1) in the figure); and to acquire attributes that capture quantifiable properties of those classes, by mining query logs based on the class instances acquired from the documents, while guiding the extraction based on a few attributes provided as seed examples (phase (2)).

As described in Figure 2, the algorithm for deriving labeled sets of class instances starts with the acquisition of candidate pairs  $\{\mathcal{M}_E\}$  of a class label and an instance, by applying a few extraction patterns to unstructured text within Web documents  $\{\mathcal{D}\}$ , while guiding the extraction by the contents of query logs  $\{\mathcal{Q}\}$  (Step 1 in Figure 2). This is fol-

Input: set of Is-A extraction patterns  $\{\mathcal{E}\}$   
 . large repository of search queries  $\{\mathcal{Q}\}$   
 . large repository of Web docs  $\{\mathcal{D}\}$   
 . weighting parameters  $\mathcal{T} \in [0,1]$  and  $\mathcal{K} \in \overline{1..∞}$   
 Output: set of pairs of a class label and an instance  $\{\langle \mathcal{C}, \mathcal{I} \rangle\}$   
 Variables:  $\{\mathcal{S}\}$  = clusters of distributionally similar phrases  
 .  $\{\mathcal{V}\}$  = vectors of contextual matches of queries in text  
 .  $\{\mathcal{M}_E\}$  = set of pairs of a class label and an instance  
 .  $\{\mathcal{C}_S\}$  = set of class labels  
 .  $\{\mathcal{X}\}, \{\mathcal{Y}\}$  = sets of queries

Steps:

01.  $\{\mathcal{M}_E\}$  = Match patterns  $\{\mathcal{E}\}$  in docs  $\{\mathcal{D}\}$  around  $\{\mathcal{Q}\}$
02.  $\{\mathcal{V}\}$  = Match phrases  $\{\mathcal{Q}\}$  in docs  $\{\mathcal{D}\}$
03.  $\{\mathcal{S}\}$  = Generate clusters of queries based on vectors  $\{\mathcal{V}\}$
04. For each cluster of phrases  $\mathcal{S}$  in  $\{\mathcal{S}\}$
05.  $\{\mathcal{C}_S\} = \emptyset$
06. For each query  $\mathcal{Q}$  of  $\mathcal{S}$
07. Insert labels of  $\mathcal{Q}$  from  $\{\mathcal{M}_E\}$  into  $\{\mathcal{C}_S\}$
08. For each label  $\mathcal{C}_S$  of  $\{\mathcal{C}_S\}$
09.  $\{\mathcal{X}\}$  = Find queries of  $\mathcal{S}$  with the label  $\mathcal{C}_S$  in  $\{\mathcal{M}_E\}$
10.  $\{\mathcal{Y}\}$  = Find clusters of  $\{\mathcal{S}\}$  containing some query with the label  $\mathcal{C}_S$  in  $\{\mathcal{M}_E\}$
11. If  $|\{\mathcal{X}\}| > \mathcal{T} \times |\{\mathcal{S}\}|$
12. If  $|\{\mathcal{Y}\}| < \mathcal{K}$
13. For each query  $\mathcal{X}$  of  $\{\mathcal{X}\}$
14. Insert pair  $\langle \mathcal{C}_S, \mathcal{X} \rangle$  into output pairs  $\{\langle \mathcal{C}, \mathcal{I} \rangle\}$
15. Return pairs  $\{\langle \mathcal{C}, \mathcal{I} \rangle\}$

Figure 2: Acquisition of labeled sets of class instances

lowed by the generation of unlabeled clusters  $\{\mathcal{S}\}$  of distributionally similar queries, by clustering vectors of contextual features collected around the occurrences of queries  $\{\mathcal{Q}\}$  within documents  $\{\mathcal{D}\}$  (Steps 2 and 3). Finally, the intermediate data  $\{\mathcal{M}_E\}$  and  $\{\mathcal{S}\}$  is merged and filtered into smaller, more accurate labeled sets of instances (Steps 4 through 15).

Step 1 in Figure 2 applies lexico-syntactic patterns  $\{\mathcal{E}\}$  that aim at extracting Is-A pairs of an instance (e.g., *Google*) and an associated class label (e.g., *Internet search engines*) from text. The two patterns, which are inspired by (Hearst, 1992) and have been the de-facto extraction technique in previous work on extracting conceptual hierarchies from text (cf. (Ponzetto and Strube, 2007; Snow et al., 2006)), can be summarized as:

$\{[.] \mathcal{C} [\text{such as}|\text{including}] \mathcal{I} [\text{and},|,]\}$ ,

where  $\mathcal{I}$  is a potential instance (e.g., *Venezuelan equine encephalitis*) and  $\mathcal{C}$  is a potential class label for the instance (e.g., *zoonotic diseases*), for example in the sentence: “*The expansion of the farms increased the spread of zoonotic diseases such as Venezuelan equine encephalitis [.]*”.

During matching, all string comparisons are case-insensitive. In order for a pattern to match a sentence, two conditions must be met. First, the class



label  $\mathcal{C}$  from the sentence must be a non-recursive noun phrase whose last component is a plural-form noun (e.g., *zoonotic diseases* in the above sentence). Second, the instance  $\mathcal{I}$  from the sentence must also occur as a complete query somewhere in the query logs  $\{\mathcal{Q}\}$ , that is, a query containing the instance and nothing else. This heuristic acknowledges the difficulty of pinpointing complex entities within documents (Downey et al., 2007), and embodies the hypothesis that, if an instance is prominent, Web search users will eventually ask about it.

In Steps 4 through 14 from Figure 2, each cluster is inspected by scanning all labels attached to one or more queries from the cluster. For each label  $\mathcal{C}_S$ , if a)  $\{\mathcal{M}_E\}$  indicates that a large number of all queries from the cluster are attached to the label (as controlled by the parameter  $\mathcal{J}$  in Step 12); and b) those queries are a significant portion of all queries from all clusters attached to the same label in  $\{\mathcal{M}_E\}$  (as controlled by the parameter  $\mathcal{K}$  in Step 13), then the label  $\mathcal{C}_S$  and each query with that label are stored in the output pairs  $\{\langle \mathcal{C}, \mathcal{I} \rangle\}$  (Steps 13 and 14). The parameters  $\mathcal{J}$  and  $\mathcal{K}$  can be used to emphasize precision (higher  $\mathcal{J}$  and lower  $\mathcal{K}$ ) or recall (lower  $\mathcal{J}$  and higher  $\mathcal{K}$ ). The resulting pairs of an instance and a class label are arranged into sets of class instances (e.g.,  $\{\textit{rabies, west nile virus, leptospirosis, ...}\}$ ), each associated with a class label (e.g., *zoonotic diseases*), and returned in Step 15.

## 2.2 Open-Domain Class Attributes

The labeled classes of instances collected automatically from Web documents are passed as input to phase (2) from Figure 1, which acquires class attributes by mining a collection of Web search queries. The attributes capture properties that are relevant to the class. The extraction of attributes exploits the set of class instances rather than the associated class label, and consists of four stages:

- 1) identification of a noisy pool of candidate attributes, as remainders of queries that also contain one of the class instances. In the case of the class *movies*, whose instances include *jay and silent bob strike back* and *kill bill*, the query “*cast jay and silent bob strike back*” produces the candidate attribute *cast*;

- 2) construction of internal search-signature vector representations for each candidate attribute, based

on queries (e.g., “*cast selection for kill bill*”) that contain a candidate attribute (*cast*) and a class instance (*kill bill*). These vectors consist of counts tied to the frequency with which an attribute occurs with a given “templated” query. The latter replaces specific attributes and instances from the query with common placeholders, e.g., “*X for Y*”;

- 3) construction of a reference internal search-signature vector representation for a small set of seed attributes provided as input. A reference vector is the normalized sum of the individual vectors corresponding to the seed attributes;

- 4) ranking of candidate attributes with respect to each class (e.g., *movies*), by computing similarity scores between their individual vector representations and the reference vector of the seed attributes.

The result of the four stages is a ranked list of attributes (e.g.,  $[\textit{opening song, cast, characters, ...}]$ ) for each class (e.g., *movies*).

In a departure from previous work, the instances of each input class are automatically generated as described earlier, rather than manually assembled. Furthermore, the amount of supervision is limited to seed attributes being provided for only one of the classes, whereas (Paşca, 2007) requires seed attributes for each class. To this effect, the extraction includes modifications such that only one reference vector is constructed internally from the seed attributes during the third stage, rather one such vector for each class in (Paşca, 2007); and similarity scores are computed cross-class by comparing vector representations of individual candidate attributes against the only reference vector available during the fourth stage, rather than with respect to the reference vector of each class in (Paşca, 2007).

## 3 Evaluation

### 3.1 Textual Data Sources

The acquisition of open-domain knowledge, in the form of class instances, labels and attributes, relies on unstructured text available within Web documents maintained by, and search queries submitted to, the Google search engine.

The collection of queries is a random sample of fully-anonymized queries in English submitted by Web users in 2006. The sample contains approximately 50 million unique queries. Each query is

Found in WordNet?	Count	Pct.	Examples
Yes (original)	1931	42.2%	baseball players, endangered species
Yes (removal)	2614	57.0%	caribbean countries, fundamental rights
No	38	0.8%	agrochemicals, celebs, handhelds, mangas

Table 1: Class labels found in WordNet in original form, or found in WordNet after removal of leading words, or not found in WordNet at all

accompanied by its frequency of occurrence in the logs. The document collection consists of approximately 100 million Web documents in English, as available in a Web repository snapshot from 2006. The textual portion of the documents is cleaned of HTML, tokenized, split into sentences and part-of-speech tagged using the TnT tagger (Brants, 2000).

### 3.2 Evaluation of Labeled Classes of Instances

**Extraction Parameters:** The set of instances that can be potentially acquired by the extraction algorithm described in Section 2.1 is heuristically limited to the top five million queries with the highest frequency within the input query logs. In the extracted data, a class label (e.g., *search engines*) is associated with one or more instances (e.g., *google*). Similarly, an instance (e.g., *google*) is associated with one or more class labels (e.g., *search engines* and *internet search engines*). The values chosen for the weighting parameters  $\mathcal{J}$  and  $\mathcal{K}$  from Section 2.1 are 0.01 and 30 respectively. After discarding classes with fewer than 25 instances, the extracted set of classes consists of 4,583 class labels, each of them associated with 25 to 7,967 instances, with an average of 189 instances per class.

**Accuracy of Class Labels:** Built over many years of manual construction efforts, lexical gold standards such as WordNet (Fellbaum, 1998) provide wide-coverage upper ontologies of the English language. Built-in morphological normalization routines make it straightforward to verify whether a class label (e.g., *faculty members*) exists as a concept in WordNet (e.g., *faculty member*). When an extracted label (e.g., *central nervous system disorders*) is not found in WordNet, it is looked up again after iteratively removing its leading words (e.g., *nervous system dis-*

Class Label={Set of Instances}	Parent in WordNet	C?
american composers={aaron copland, eric ewazen, george gershwin,...}	composers	Y
modern appliances={built-in oven, ceramic hob, tumble dryer,...}	appliances	S
area hospitals={carolinas medical center, nyack hospital,...}	hospitals	S
multiple languages={chuukese, ladino, mandarin, us english,...}	languages	N

Table 2: Correctness judgments for extracted classes whose class labels are found in WordNet only after removal of their leading words (C=Correctness, Y=correct, S=subjectively correct, N=incorrect)

*orders, system disorders and disorders*).

As shown in Table 1, less than half of the 4,583 extracted class labels (e.g., *baseball players*) are found in their original forms in WordNet. The majority of the class labels (2,614 out of 4,583) can be found in WordNet only after removal of one or more leading words (e.g., *caribbean countries*), which suggests that many of the class labels correspond to finer-grained, automatically-extracted concepts that are not available in the manually-built WordNet. To test whether that is the case, a random sample of 200 class labels, out of the 2,614 labels found to be potentially-useful specific concepts, are manually annotated as correct, subjectively correct or incorrect, as shown in Table 2. A class label is: correct, if it captures a relevant concept although it could not be found in WordNet; subjectively correct, if it is relevant not in general but only in a particular context, either from a subjective viewpoint (e.g., *modern appliances*), or relative to a particular temporal anchor (e.g., *current players*), or in connection to a particular geographical area (e.g., *area hospitals*); or incorrect, if it does not capture any useful concept (e.g., *multiple languages*). The manual analysis of the sample of 200 class labels indicates that 154 (77%) are relevant concepts and 27 (13.5%) are subjectively relevant concepts, for a total of 181 (90.5%) relevant concepts, whereas 19 (9.5%) of the labels are incorrect. It is worth emphasizing the importance of automatically-collected classes judged as relevant and not present in WordNet: *caribbean countries, computer manufacturers, entertainment companies, market research firms* are arguably very useful and should probably be considered as part of

Class Label		Size of Instance Sets			Class Label		Size of Instance Sets		
$M$ (Manual)	$E$ (Extracted)	$M$	$E$	$\frac{M \cap E}{M}$	$M$ (Manual)	$E$ (Extracted)	$M$	$E$	$\frac{M \cap E}{M}$
Actor	actors	1500	696	23.73	Movie	movies	626	2201	30.83
AircraftModel	-	217	-	-	NationalPark	parks	59	296	0
Award	awards	200	283	13	NbaTeam	nba teams	30	66	86.66
BasicFood	foods	155	3484	61.93	Newspaper	newspapers	599	879	16.02
CarModel	car models	368	48	5.16	Painter	painters	1011	823	22.45
CartoonChar	cartoon characters	50	144	36	ProgLanguage	programming languages	101	153	26.73
CellPhoneModel	cell phones	204	49	0	Religion	religions	128	72	11.71
ChemicalElem	chemicals	118	487	1.69	River	river systems	167	118	15.56
City	cities	589	3642	50.08	SearchEngine	search engines	25	133	64
Company	companies	738	7036	26.01	SkyBody	constellations	97	37	1.03
Country	countries	197	677	91.37	Skyscraper	-	172	-	-
Currency	currencies	55	128	25.45	SoccerClub	football clubs	116	101	22.41
DigitalCamera	digital cameras	534	58	0.18	SportEvent	sports events	143	73	12.58
Disease	diseases	209	3566	65.55	Stadium	stadiums	190	92	6.31
Drug	drugs	345	1209	44.05	TerroristGroup	terrorist groups	74	134	33.78
Empire	empires	78	54	6.41	Treaty	treaties	202	200	7.42
Flower	flowers	59	642	25.42	University	universities	501	1127	21.55
Holiday	holidays	82	300	48.78	VideoGame	video games	450	282	17.33
Hurricane	-	74	-	-	Wine	wines	60	270	56.66
Mountain	mountains	245	49	7.75	WorldWarBattle	battles	127	135	9.44
Total mapped: 37 out of 40 classes							-	-	<b>26.89</b>

Table 3: Comparison between manually-assembled instance sets of gold-standard classes ( $M$ ) and instance sets of automatically-extracted classes ( $E$ ). Each gold-standard class ( $M$ ) was manually mapped into an extracted class ( $E$ ), unless no relevant mapping was found. Ratios ( $\frac{M \cap E}{M}$ ) are shown as percentages

any refinements to hand-built hierarchies, including any future extensions of WordNet.

**Accuracy of Class Instances:** The computation of the precision of the extracted instances (e.g., *fifth element* and *kill bill* for the class label *movies*) relies on manual inspection of all instances associated to a sample of the extracted class labels. Rather than inspecting a random sample of classes, the evaluation validates the results against a reference set of 40 gold-standard classes that were manually assembled as part of previous work (Paşca, 2007). A class from the gold standard consists of a manually-created class label (e.g., *AircraftModel*) associated with a manually-assembled, and therefore high-precision, set of representative instances of the class.

To evaluate the precision of the extracted instances, the manual label of each gold-standard class (e.g., *SearchEngine*) is mapped into a class label extracted from text (e.g., *search engines*). As shown in the first two columns of Table 3, the mapping into extracted class labels succeeds for 37 of the 40 gold-standard classes. 28 of the 37 mappings involve linking an abstract class label (e.g., *SearchEngine*)

with the corresponding plural forms among the extracted class labels (e.g., *search engines*). The remaining 9 mappings link a manual class label with either an equivalent extracted class label (e.g., *SoccerClub* with *football clubs*), or a strongly-related class label (e.g., *NationalPark* with *parks*). No mapping is found for 3 out of the 40 classes, namely *AircraftModel*, *Hurricane* and *Skyscraper*, which are therefore removed from consideration.

The sizes of the instance sets available for each class in the gold standard are compared in the third through fifth columns of Table 3. In the table,  $M$  stands for manually-assembled instance sets, and  $E$  for automatically-extracted instance sets. For example, the gold-standard class *SearchEngine* contains 25 manually-collected instances, while the parallel class label *search engines* contains 133 automatically-extracted instances. The fifth column shows the percentage of manually-collected instances ( $M$ ) that are also extracted automatically ( $E$ ). In the case of the class *SearchEngine*, 16 of the 25 manually-collected instances are among the 133 automatically-extracted instances of the same class,

Label	Value	Examples of Attributes
vital	1.0	<i>investors</i> : investment strategies
okay	0.5	<i>religious leaders</i> : coat of arms
wrong	0.0	<i>designers</i> : stephanie

Table 4: Labels for assessing attribute correctness

which corresponds to a relative coverage of 64% of the manually-collected instance set. Some instances may occur within the manually-collected set but not the automatically-extracted set (e.g., *zoom-info* and *brainbot* for the class *SearchEngine*) or, more frequently, vice-versa (e.g., *surfwax*, *blinkx*, *entireweb*, *web wombat*, *exalead* etc.). Overall, the relative coverage of automatically-extracted instance sets with respect to manually-collected instance sets is 26.89%, as an average over the 37 gold-standard classes. More significantly, the size advantage of automatically-extracted instance sets is not the undesirable result of those sets containing many spurious instances. Indeed, the manual inspection of the automatically-extracted instances sets indicates an average accuracy of 79.3% over the 37 gold-standard classes retained in the experiments. To summarize, the method proposed in this paper acquires open-domain classes from unstructured text of arbitrary quality, without a-priori restrictions to specific domains of interest and with virtually no supervision (except for the ubiquitous Is-A extraction patterns), at accuracy levels of around 90% for class labels and 80% for class instances.

### 3.3 Evaluation of Class Attributes

**Extraction Parameters:** Given a target class specified as a set of instances and a set of five seed attributes for a class (e.g.,  $\{quality, speed, number\ of\ users, market\ share, reliability\}$  for *SearchEngine*), the method described in Section 2.2 extracts ranked lists of class attributes from the input query logs. Internally, the ranking uses Jensen-Shannon (Lee, 1999) to compute similarity scores between internal representations of seed attributes, on one hand, and each of the candidate attributes, on the other hand.

**Evaluation Procedure:** To remove any possible bias towards higher-ranked attributes during the assessment of class attributes, the ranked lists of attributes to be evaluated are sorted alphabetically into a merged list. Each attribute of the merged list is

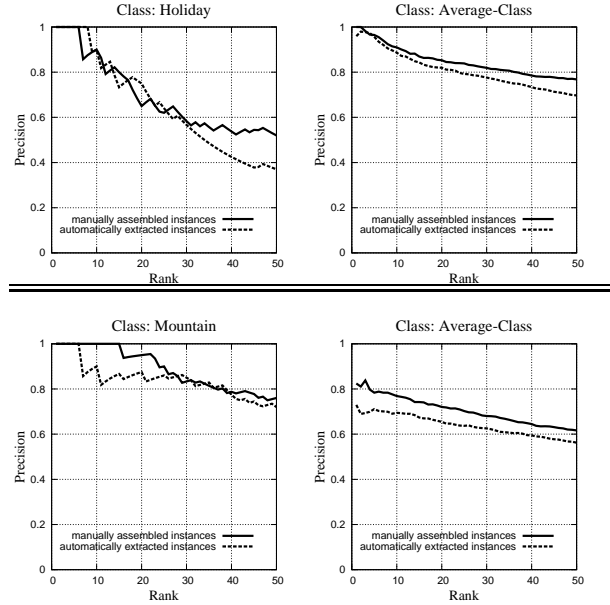


Figure 3: Accuracy of attributes extracted based on manually assembled, gold standard ( $M$ ) vs. automatically extracted ( $E$ ) instance sets, for a few target classes (left-most graphs) and as an average over all (37) target classes (right-most graphs). Seed attributes are provided as input for each target class (top graphs), or for only one target class (bottom graphs)

manually assigned a correctness label within its respective class. An attribute is *vital* if it must be present in an ideal list of attributes of the class; *okay* if it provides useful but non-essential information; and *wrong* if it is incorrect.

To compute the overall precision score over a ranked list of extracted attributes, the correctness labels are converted to numeric values as shown in Table 4. Precision at some rank  $N$  in the list is thus measured as the sum of the assigned values of the first  $N$  candidate attributes, divided by  $N$ .

**Accuracy of Class Attributes:** Figure 3 plots precision values for ranks 1 through 50 of the lists of attributes extracted through several runs over the 37 gold-standard classes described in the previous section. The runs correspond to different amounts of supervision, specified through a particular choice in the number of seed attributes, and in the source of instances passed as input to the system:

- number of input seed attributes: seed attributes are provided either for each of the 37 classes, for a total of  $5 \times 37 = 185$  attributes (the graphs at the top of Figure 3); or only for one class (namely, *Country*),

Class		Precision				Top Ten Extracted Attributes
#	Class Label={Set of Instances}	@5	@10	@15	@20	
1	accounting systems={flexcube, myob, oracle fi nancials, peachtree accounting, sybiz,...}	0.70	0.70	0.77	0.70	overview, architecture, interview questions, free downloads, canadian version, passwords, modules, crystal reports, property management, free trial
2	antimicrobials={azithromycin, chloramphenicol, fusidic acid, quinolones, sulfa drugs,...}	1.00	1.00	0.93	0.95	chemical formula, chemical structure, history, invention, inventor, defi nition, mechanism of action, side-effects, uses, shelf life
5	civilizations={ancient greece, chaldeans, etruscans, inca indians, roman republic,...}	1.00	1.00	0.93	0.90	social pyramid, climate, geography, flag, population, social structure, natural resources, family life, god, goddesses
9	farm animals={angora goats, burros, cattle, cows, donkeys, draft horses, mule, oxen,...}	1.00	0.80	0.83	0.80	digestive system, evolution, domestication, gestation period, scientifi c name, adaptations, coloring pages, p**, body parts, selective breeding
10	forages={alsike clover, rye grass, tall fescue, sericea lespedeza,...}	0.90	0.95	0.73	0.57	types, picture, weed control, planting, uses, information, herbicide, germination, care, fertilizer
Average-Class (25 classes)		<b>0.75</b>	<b>0.70</b>	<b>0.68</b>	<b>0.67</b>	

Table 5: Precision of attributes extracted for a sample of 25 classes. Seed attributes are provided for only one class.

for a total of 5 attributes over all classes (the graphs at the bottom of Figure 3);

- source of input instance sets: the instance sets for each class are either manually collected ( $M$  from Table 3), or automatically extracted ( $E$  from Table 3). The choices correspond to the two curves plotted in each graph in Figure 3.

The graphs in Figure 3 show the precision over individual target classes (leftmost graphs), and as an average over all 37 classes (rightmost graphs). As expected, the precision of the extracted attributes as an average over all classes is best when the input instance sets are hand-picked ( $M$ ), as opposed to automatically extracted ( $E$ ). However, the loss of precision from  $M$  to  $E$  is small at all measured ranks.

Table 5 offers an alternative view on the quality of the attributes extracted for a random sample of 25 classes out of the larger set of 4,583 classes acquired from text. The 25 classes are passed as input for attribute extraction without modifications. In particular, the instance sets are not manually post-filtered or otherwise changed in any way. To keep the time required to judge the correctness of all extracted attributes within reasonable limits, the evaluation considers only the top 20 (rather than 50) attributes extracted per class. As shown in Table 5, the method proposed in this paper acquires attributes for automatically-extracted, open-domain classes, without a-priori restrictions to specific domains of interest and relying on only five seed attributes specified

for only one class, at accuracy levels reaching 70% at rank 10, and 67% at rank 20.

## 4 Related Work

### 4.1 Acquisition of Classes of Instances

Although some researchers focus on re-organizing or extending classes of instances already available explicitly within manually-built resources such as Wikipedia (Ponzetto and Strube, 2007) or WordNet (Snow et al., 2006) or both (Suchanek et al., 2007), a large body of previous work focuses on compiling sets of instances, not necessarily labeled, from unstructured text. The extraction proceeds either iteratively by starting from a few seed extraction rules (Collins and Singer, 1999), or by mining named entities from comparable news articles (Shinyama and Sekine, 2004) or from multilingual corpora (Klementiev and Roth, 2006).

A bootstrapping method (Riloff and Jones, 1999) cautiously grows very small seed sets of five instances of the same class, to fewer than 300 items after 50 consecutive iterations, with a final precision varying between 46% and 76% depending on the type of semantic lexicon. Experimental results from (Feldman and Rosenfeld, 2006) indicate that named entity recognizers can boost the performance of weakly supervised extraction of class instances, but only for a few coarse-grained types such as *Person* and only if they are simpler to recognize in text (Feldman and Rosenfeld, 2006).

In (Cafarella et al., 2005), handcrafted extraction patterns are applied to a collection of 60 million Web documents to extract instances of the classes *Company* and *Country*. Based on the manual evaluation of samples of extracted instances, an estimated number of 1,116 instances of *Company* are extracted at a precision score of 90%. In comparison, the approach of this paper pursues a more aggressive goal, by extracting a larger and more diverse number of labeled classes, whose instances are often more difficult to extract than country names and most company names, at precision scores of almost 80%.

The task of extracting relevant labels to describe sets of documents, rather than sets of instances, is explored in (Treeratpituk and Callan, 2006). Given pre-existing sets of instances, (Pantel and Ravichandran, 2004) investigates the task of acquiring appropriate class labels to the sets from unstructured text. Various class labels are assigned to a total of 1,432 sets of instances. The accuracy of the class labels is computed over a sample of instances, by manually assessing the correctness of the top five labels returned by the system for each instance. The resulting mean reciprocal rank of 77% gives partial credit to labels of an evaluated instance, even if only the fourth or fifth assigned labels are correct. Our evaluation of the accuracy of class labels is stricter, as it considers only one class label of a given instance at a time, rather than a pool of the best candidate labels.

As a pre-requisite to extracting relations among pairs of classes, the method described in (Davidov et al., 2007) extracts class instances from unstructured Web documents, by submitting pairs of instances as queries and analyzing the contents of the top 1,000 documents returned by a Web search engine. For each target class, a small set of instances must be provided manually as seeds. As such, the method can be applied to the task of extracting a large set of open-domain classes only after manually enumerating through the entire set of target classes, and providing seed instances for each. Furthermore, no attempt is made to extract relevant class labels for the sets of instances. Comparatively, the open-domain classes extracted in our paper have an explicit label in addition to the sets of instances, and do not require identifying the range of the target classes in advance, or providing any seed instances as input. The evaluation methodology is also quite dif-

ferent, as the instance sets acquired based on the input seed instances in (Davidov et al., 2007) are only evaluated for three hand-picked classes, with precision scores of 90% for names of *countries*, 87% for *fish species* and 68% for instances of *constellations*. Our evaluation of the accuracy of class instances is again stricter, since the evaluation sample is larger, and includes more varied classes, whose instances are sometimes more difficult to identify in text.

## 4.2 Acquisition of Class Attributes

Previous work on the automatic acquisition of attributes for open-domain classes from text is less general than the extraction method and experiments presented in our paper. Indeed, previous evaluations were restricted to small sets of classes (forty classes in (Paşca, 2007)), whereas our evaluations also consider a random, more diverse sample of open-domain classes. More importantly, by dropping the requirement of manually providing a small set of seed attributes for each target class, and relying on only a few seed attributes specified for one reference class, we harvest class attributes without the need of first determining what the classes should be, what instances they should contain, and from which resources the instances should be collected.

## 5 Conclusion

In a departure from previous approaches to large-scale information extraction from unstructured text on the Web, this paper introduces a weakly-supervised extraction framework for mining useful knowledge from a combination of both documents and search query logs. In evaluations over labeled classes of instances extracted without a-priori restrictions to specific domains of interest and with very little supervision, the accuracy exceeds 90% for class labels, approaches 80% for class instances, and exceeds 70% (at rank 10) and 67% (at rank 20) for class attributes. Current work aims at expanding the number of instances within each class while retaining similar precision levels; extracting attributes with more consistent precision scores across classes from different domains; and introducing confidence scores in attribute extraction, allowing for the detection of classes for which it is unlikely to extract large numbers of useful attributes from text.

## References

- M. Banko, Michael J Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2670–2676, Hyderabad, India.
- T. Brants. 2000. TnT - a statistical part of speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP-00)*, pages 224–231, Seattle, Washington.
- M. Cafarella, D. Downey, S. Soderland, and O. Etzioni. 2005. KnowItNow: Fast, scalable information extraction from the Web. In *Proceedings of the Human Language Technology Conference (HLT-EMNLP-05)*, pages 563–570, Vancouver, Canada.
- M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the 1999 Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, pages 189–196, College Park, Maryland.
- D. Davidov, A. Rappoport, and M. Koppel. 2007. Fully unsupervised discovery of concept-specific relationships by Web mining. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 232–239, Prague, Czech Republic.
- D. Downey, M. Broadhead, and O. Etzioni. 2007. Locating complex named entities in Web text. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2733–2739, Hyderabad, India.
- R. Feldman and B. Rosenfeld. 2006. Boosting unsupervised relation extraction by using NER. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP-ACL-06)*, pages 473–481, Sydney, Australia.
- C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press.
- M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539–545, Nantes, France.
- A. Klementiev and D. Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 817–824, Sydney, Australia.
- L. Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association of Computational Linguistics (ACL-99)*, pages 25–32, College Park, Maryland.
- D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-98)*, pages 768–774, Montreal, Quebec.
- R. Mooney and R. Bunescu. 2005. Mining knowledge from text using information extraction. *SIGKDD Explorations*, 7(1):3–10.
- M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. 2006. Organizing and searching the World Wide Web of facts - step one: the one-million fact extraction challenge. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 1400–1405, Boston, Massachusetts.
- M. Paşca. 2007. Organizing and searching the World Wide Web of facts - step two: Harnessing the wisdom of the crowds. In *Proceedings of the 16th World Wide Web Conference (WWW-07)*, pages 101–110, Banff, Canada.
- P. Pantel and D. Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of the 2004 Human Language Technology Conference (HLT-NAACL-04)*, pages 321–328, Boston, Massachusetts.
- S. Ponzetto and M. Strube. 2007. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI-07)*, pages 1440–1447, Vancouver, British Columbia.
- E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 474–479, Orlando, Florida.
- Y. Shinyama and S. Sekine. 2004. Named entity discovery using comparable news articles. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 848–853, Geneva, Switzerland.
- R. Snow, D. Jurafsky, and A. Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 801–808, Sydney, Australia.
- F. Suchanek, G. Kasneci, and G. Weikum. 2007. Yago: a core of semantic knowledge unifying WordNet and Wikipedia. In *Proceedings of the 16th World Wide Web Conference (WWW-07)*, pages 697–706, Banff, Canada.
- P. Treeratpituk and J. Callan. 2006. Automatically labeling hierarchical clusters. In *Proceedings of the 7th Annual Conference on Digital Government Research (DGO-06)*, pages 167–176, San Diego, California.

# The Tradeoffs Between Open and Traditional Relation Extraction

Michele Banko and Oren Etzioni

Turing Center

University of Washington

Computer Science and Engineering

Box 352350

Seattle, WA 98195, USA

banko, etzioni@cs.washington.edu

## Abstract

Traditional Information Extraction (IE) takes a relation name and hand-tagged examples of that relation as input. Open IE is a *relation-independent* extraction paradigm that is tailored to massive and heterogeneous corpora such as the Web. An Open IE system extracts a diverse set of relational tuples from text without *any* relation-specific input. How is Open IE possible? We analyze a sample of English sentences to demonstrate that numerous relationships are expressed using a compact set of relation-independent lexico-syntactic patterns, which can be learned by an Open IE system.

What are the tradeoffs between Open IE and traditional IE? We consider this question in the context of two tasks. First, when the number of relations is massive, and the relations themselves are not pre-specified, we argue that Open IE is necessary. We then present a new model for Open IE called O-CRF and show that it achieves increased precision and nearly double the recall than the model employed by TEXTRUNNER, the previous state-of-the-art Open IE system. Second, when the number of target relations is small, and their names are known in advance, we show that O-CRF is able to match the precision of a traditional extraction system, though at substantially lower recall. Finally, we show how to combine the two types of systems into a hybrid that achieves higher precision than a traditional extractor, with comparable recall.

## 1 Introduction

Relation Extraction (RE) is the task of recognizing the assertion of a particular relationship between two or more entities in text. Typically, the target relation (*e.g.*, seminar location) is given to the RE system as input along with hand-crafted extraction patterns or patterns learned from hand-labeled training examples (Brin, 1998; Riloff and Jones, 1999; Agichtein and Gravano, 2000). Such inputs are *specific* to the target relation. Shifting to a new relation requires a person to manually create new extraction patterns or specify new training examples. This manual labor scales linearly with the number of target relations.

In 2007, we introduced a new approach to the RE task, called Open Information Extraction (Open IE), which scales RE to the Web. An Open IE system extracts a diverse set of relational tuples without requiring *any* relation-specific human input. Open IE's extraction process is linear in the number of documents in the corpus, and constant in the number of relations. Open IE is ideally suited to corpora such as the Web, where the target relations are not known in advance, and their number is massive.

The relationship between standard RE systems and the new Open IE paradigm is analogous to the relationship between lexicalized and unlexicalized parsers. Statistical parsers are usually *lexicalized* (*i.e.* they make parsing decisions based on n-gram statistics computed for specific lexemes). However, Klein and Manning (2003) showed that *unlexicalized* parsers are more accurate than previously believed, and can be learned in an unsupervised manner. Klein and Manning analyze the tradeoffs be-



tween the two approaches to parsing and argue that state-of-the-art parsing will benefit from employing both approaches in concert. In this paper, we examine the tradeoffs between relation-specific (“lexicalized”) extraction and relation-independent (“unlexicalized”) extraction and reach an analogous conclusion.

Is it, in fact, possible to learn relation-independent extraction patterns? What do they look like? We first consider the task of open extraction, in which the goal is to extract relationships from text when their number is large and identity unknown. We then consider the targeted extraction task, in which the goal is to locate instances of a known relation. How does the precision and recall of Open IE compare with that of relation-specific extraction? Is it possible to combine Open IE with a “lexicalized” RE system to improve performance? This paper addresses the questions raised above and makes the following contributions:

- We present O-CRF, a new Open IE system that uses Conditional Random Fields, and demonstrate its ability to extract a variety of relations with a precision of 88.3% and recall of 45.2%. We compare O-CRF to O-NB, the extraction model previously used by *TEXTRUNNER* (Banko et al., 2007), a state-of-the-art Open IE system. We show that O-CRF achieves a relative gain in F-measure of 63% over O-NB.
- We provide a corpus-based characterization of how binary relationships are expressed in English to demonstrate that learning a relation-independent extractor is feasible, at least for the English language.
- In the targeted extraction case, we compare the performance of O-CRF to a traditional RE system and find that without any relation-specific input, O-CRF obtains the same precision with lower recall compared to a lexicalized extractor trained using hundreds, and sometimes thousands, of labeled examples per relation.
- We present H-CRF, an ensemble-based extractor that learns to combine the output of the lexicalized and unlexicalized RE systems and achieves a 10% relative increase in precision with comparable recall over traditional RE.

The remainder of this paper is organized as follows. Section 2 assesses the promise of relation-independent extraction for the English language by characterizing how a sample of relations is expressed in text. Section 3 describes O-CRF, a new Open IE system, as well as R1-CRF, a standard RE system; a hybrid RE system is then presented in Section 4. Section 5 reports on our experimental results. Section 6 considers related work, which is then followed by a discussion of future work.

## 2 The Nature of Relations in English

How are relationships expressed in English sentences? In this section, we show that many relationships are consistently expressed using a compact set of relation-independent lexico-syntactic patterns, and quantify their frequency based on a sample of 500 sentences selected at random from an IE training corpus developed by (Bunescu and Mooney, 2007).<sup>1</sup> This observation helps to explain the success of open relation extraction, which learns a relation-independent extraction model as described in Section 3.1.

Previous work has noted that distinguished relations, such as hypernymy (is-a) and meronymy (part-whole), are often expressed using a small number of lexico-syntactic patterns (Hearst, 1992). The manual identification of these patterns inspired a body of work in which this initial set of extraction patterns is used to seed a bootstrapping process that automatically acquires additional patterns for is-a or part-whole relations (Etzioni et al., 2005; Snow et al., 2005; Girju et al., 2006). It is quite natural then to consider whether the same can be done for *all* binary relationships.

To characterize how binary relationships are expressed, one of the authors of this paper carefully studied the labeled relation instances and produced a lexico-syntactic pattern that captured the relation for each instance. Interestingly, we found that 95% of the patterns could be grouped into the categories listed in Table 1. Note, however, that the patterns shown in Table 1 are greatly simplified by omitting the exact conditions under which they will reliably produce a correct extraction. For instance, while many relationships are indicated strictly by a verb,

<sup>1</sup>For simplicity, we restrict our study to binary relationships.

Relative Frequency	Category	Simplified Lexico-Syntactic Pattern
37.8	Verb	$E_1$ Verb $E_2$ <i>X established Y</i>
22.8	Noun+Prep	$E_1$ NP Prep $E_2$ <i>X settlement with Y</i>
16.0	Verb+Prep	$E_1$ Verb Prep $E_2$ <i>X moved to Y</i>
9.4	Infinitive	$E_1$ to Verb $E_2$ <i>X plans to acquire Y</i>
5.2	Modifier	$E_1$ Verb $E_2$ Noun <i>X is Y winner</i>
1.8	Coordinate <sub>n</sub>	$E_1$ (and , - :) $E_2$ NP <i>X-Y deal</i>
1.0	Coordinate <sub>v</sub>	$E_1$ (and ,) $E_2$ Verb <i>X, Y merge</i>
0.8	Appositive	$E_1$ NP (: ,)? $E_2$ <i>X hometown : Y</i>

Table 1: Taxonomy of Binary Relationships: Nearly 95% of 500 randomly selected sentences belongs to one of the eight categories above.

detailed contextual cues are required to determine, exactly which, if any, verb observed in the context of two entities is indicative of a relationship between them. In the next section, we show how we can use a Conditional Random Field, a model that can be described as a finite state machine with weighted transitions, to learn a model of how binary relationships are expressed in English.

### 3 Relation Extraction

Given a relation name, labeled examples of the relation, and a corpus, traditional Relation Extraction (RE) systems output instances of the given relation found in the corpus. In the open extraction task, relation names are not known in advance. The sole input to an Open IE system is a corpus, along with a small set of relation-independent heuristics, which are used to learn a general model of extraction for all relations at once.

The task of open extraction is notably more difficult than the traditional formulation of RE for several reasons. First, traditional RE systems do not attempt to extract the text that signifies a relation in a sentence, since the relation name is given. In con-

trast, an Open IE system has to locate both the set of entities believed to participate in a relation, and the salient textual cues that indicate the relation among them. Knowledge extracted by an open system takes the form of relational tuples  $(r, e_1, \dots, e_n)$  that contain two or more entities  $e_1, \dots, e_n$ , and  $r$ , the name of the relationship among them. For example, from the sentence, “Microsoft is headquartered in beautiful Redmond”, we expect to extract  $(is\ headquartered\ in, Microsoft, Redmond)$ . Moreover, following extraction, the system must identify exactly which relation strings  $r$  correspond to a general relation of interest. To ensure high-levels of coverage on a *per-relation* basis, we need, for example to deduce that “’s headquarters in”, “is headquartered in” and “is based in” are different ways of expressing HEADQUARTERS(X,Y).

Second, a relation-independent extraction process makes it difficult to leverage the full set of features typically used when performing extraction one relation at a time. For instance, the presence of the words *company* and *headquarters* will be useful in detecting instances of the HEADQUARTERS(X,Y) relation, but are not useful features for identifying relations in general. Finally, RE systems typically use named-entity types as a guide (e.g., the second argument to HEADQUARTERS should be a LOCATION). In Open IE, the relations are not known in advance, and neither are their argument types.

The unique nature of the open extraction task has led us to develop O-CRF, an open extraction system that uses the power of graphical models to identify relations in text. The remainder of this section describes O-CRF, and compares it to the extraction model employed by TEXTRUNNER, the first Open IE system (Banko et al., 2007). We then describe R1-CRF, a RE system that can be applied in a typical one-relation-at-a-time setting.

#### 3.1 Open Extraction with Conditional Random Fields

TEXTRUNNER initially treated Open IE as a classification problem, using a Naive Bayes classifier to predict whether heuristically-chosen tokens between two entities indicated a relationship or not. For the remainder of this paper, we refer to this model as O-NB. Whereas classifiers predict the label of a single variable, graphical models model multiple, in-

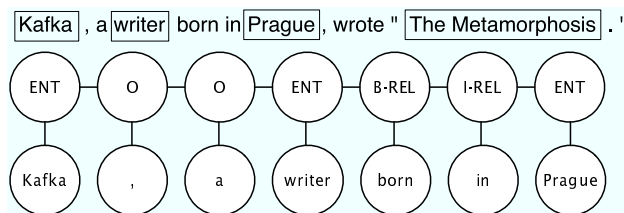


Figure 1: Relation Extraction as Sequence Labeling: A CRF is used to identify the relationship, *born in*, between *Kafka* and *Prague*

terdependent variables. Conditional Random Fields (CRFs) (Lafferty et al., 2001), are undirected graphical models trained to maximize the conditional probability of a finite set of labels  $Y$  given a set of input observations  $X$ . By making a first-order Markov assumption about the dependencies among the output variables  $Y$ , and arranging variables sequentially in a linear chain, RE can be treated as a sequence labeling problem. Linear-chain CRFs have been applied to a variety of sequential text processing tasks including named-entity recognition, part-of-speech tagging, word segmentation, semantic role identification, and recently relation extraction (Culotta et al., 2006).

### 3.1.1 Training

As with O-NB, O-CRF's training process is self-supervised. O-CRF applies a handful of relation-independent heuristics to the PennTreebank and obtains a set of labeled examples in the form of relational tuples. The heuristics were designed to capture dependencies typically obtained via syntactic parsing and semantic role labelling. For example, a heuristic used to identify positive examples is the extraction of noun phrases participating in a subject-verb-object relationship, e.g., "<Einstein> received <the Nobel Prize> in 1921." An example of a heuristic that locates negative examples is the extraction of objects that cross the boundary of an adverbial clause, e.g. "He studied <Einstein's work> when visiting <Germany>."

The resulting set of labeled examples are described using features that can be extracted without syntactic or semantic analysis and used to train a CRF, a sequence model that learns to identify spans of tokens believed to indicate explicit mentions of relationships between entities.

O-CRF first applies a phrase chunker to each document, and treats the identified noun phrases as candidate entities for extraction. Each pair of entities appearing no more than a maximum number of words apart and their surrounding context are considered as possible evidence for RE. The entity pair serves to anchor each end of a linear-chain CRF, and both entities in the pair are assigned a fixed label of ENT. Tokens in the surrounding context are treated as possible textual cues that indicate a relation, and can be assigned one of the following labels: B-REL, indicating the start of a relation, I-REL, indicating the continuation of a predicted relation, or O, indicating the token is not believed to be part of an explicit relationship. An illustration is given in Figure 1.

The set of features used by O-CRF is largely similar to those used by O-NB and other state-of-the-art relation extraction systems. They include part-of-speech tags (predicted using a separately trained maximum-entropy model), regular expressions (e.g. detecting capitalization, punctuation, etc.), context words, and conjunctions of features occurring in adjacent positions within six words to the left and six words to the right of the current word. A unique aspect of O-CRF is that O-CRF uses context words belonging only to *closed classes* (e.g. prepositions and determiners) but not function words such as verbs or nouns. Thus, unlike most RE systems, O-CRF does not try to recognize semantic classes of entities.

O-CRF has a number of limitations, most of which are shared with other systems that perform extraction from natural language text. First, O-CRF only extracts relations that are explicitly mentioned in the text; implicit relationships that could be inferred from the text would need to be inferred from O-CRF extractions. Second, O-CRF focuses on relationships that are primarily word-based, and not indicated solely from punctuation or document-level features. Finally, relations must occur between entity names within the same sentence.

O-CRF was built using the CRF implementation provided by MALLETT (McCallum, 2002), as well as part-of-speech tagging and phrase-chunking tools available from OPENNLP.<sup>2</sup>

<sup>2</sup><http://opennlp.sourceforge.net>

### 3.1.2 Extraction

Given an input corpus, O-CRF makes a single pass over the data, and performs entity identification using a phrase chunker. The CRF is then used to label instances relations for each possible entity pair, subject to the constraints mentioned previously.

Following extraction, O-CRF applies the RESOLVER algorithm (Yates and Etzioni, 2007) to find relation synonyms, the various ways in which a relation is expressed in text. RESOLVER uses a probabilistic model to predict if two strings refer to the same item, based on relational features, in an unsupervised manner. In Section 5.2 we report that RESOLVER boosts the recall of O-CRF by 50%.

### 3.2 Relation-Specific Extraction

To compare the behavior of open, or “unlexicalized,” extraction to relation-specific, or “lexicalized” extraction, we developed a CRF-based extractor under the traditional RE paradigm. We refer to this system as R1-CRF.

Although the graphical structure of R1-CRF is the same as O-CRF R1-CRF differs in a few ways. A given relation  $R$  is specified *a priori*, and R1-CRF is trained from hand-labeled positive and negative instances of  $R$ . The extractor is also permitted to use all lexical features, and is not restricted to closed-class words as is O-CRF. Since  $R$  is known in advance, if R1-CRF outputs a tuple at extraction time, the tuple is believed to be an instance of  $R$ .

## 4 Hybrid Relation Extraction

Since O-CRF and R1-CRF have complementary views of the extraction process, it is natural to wonder whether they can be combined to produce a more powerful extractor. In many machine learning settings, the use of an ensemble of diverse classifiers during prediction has been observed to yield higher levels of performance compared to individual algorithms. We now describe an ensemble-based or *hybrid* approach to RE that leverages the different views offered by open, self-supervised extraction in O-CRF, and lexicalized, supervised extraction in R1-CRF.

### 4.1 Stacking

*Stacked generalization*, or *stacking*, (Wolpert, 1992), is an ensemble-based framework in which the goal is learn a meta-classifier from the output of several base-level classifiers. The training set used to train the meta-classifier is generated using a leave-one-out procedure: for each base-level algorithm, a classifier is trained from all but one training example and then used to generate a prediction for the left-out example. The meta-classifier is trained using the predictions of the base-level classifiers as features, and the true label as given by the training data.

Previous studies (Ting and Witten, 1999; Zenko and Dzeroski, 2002; Sigletos et al., 2005) have shown that the probabilities of each class value as estimated by each base-level algorithm are effective features when training meta-learners. Stacking was shown to be consistently more effective than voting, another popular ensemble-based method in which the outputs of the base-classifiers are combined either through majority vote or by taking the class value with the highest average probability.

### 4.2 Stacked Relation Extraction

We used the stacking methodology to build an ensemble-based extractor, referred to as H-CRF. Treating the output of an O-CRF and R1-CRF as black boxes, H-CRF learns to predict which, if any, tokens found between a pair of entities ( $e_1, e_2$ ), indicates a relationship. Due to the sequential nature of our RE task, H-CRF employs a CRF as the meta-learner, as opposed to a decision tree or regression-based classifier.

H-CRF uses the probability distribution over the set of possible labels according to each O-CRF and R1-CRF as features. To obtain the probability at each position of a linear-chain CRF, the constrained forward-backward technique described in (Culotta and McCallum, 2004) is used. H-CRF also computes the Monge Elkan distance (Monge and Elkan, 1996) between the relations predicted by O-CRF and R1-CRF and includes the result in the feature set. An additional meta-feature utilized by H-CRF indicates whether either or both base extractors return “no relation” for a given pair of entities. In addition to these numeric features, H-CRF uses a subset of the base features used by O-CRF and R1-CRF. At each

Category	O-CRF			O-NB		
	P	R	F1	P	R	F1
Verb	93.9	65.1	76.9	100	38.6	55.7
Noun+Prep	89.1	36.0	51.3	100	9.7	55.7
Verb+Prep	95.2	50.0	65.6	95.2	25.3	40.0
Infinitive	95.7	46.8	62.9	100	25.5	40.6
Other	0	0	0	0	0	0
<b>All</b>	<b>88.3</b>	<b>45.2</b>	<b>59.8</b>	86.6	23.2	36.6

Table 2: Open Extraction by Relation Category. O-CRF outperforms O-NB, obtaining nearly double its recall and increased precision. O-CRF’s gains are partly due to its lower false positive rate for relationships categorized as “Other.”

given position  $i$  between  $e_1$  and  $e_2$ , the presence of the word observed at  $i$  as a feature, as well as the presence of the part-of-speech-tag at  $i$ .

## 5 Experimental Results

The following experiments demonstrate the benefits of Open IE for two tasks: open extraction and targeted extraction.

Section 5.1, assesses the ability of O-CRF to locate instances of relationships when the number of relationships is large and their identity is unknown. We show that without any relation-specific input, O-CRF extracts binary relationships with high precision and a recall that nearly doubles that of O-NB.

Sections 5.2 and 5.3 compare O-CRF to traditional and hybrid RE when the goal is to locate instances of a small set of known target relations. We find that while single-relation extraction, as embodied by R1-CRF, achieves comparatively higher levels of recall, it takes hundreds, and sometimes thousands, of labeled examples *per relation*, for R1-CRF to approach the precision obtained by O-CRF, which is self-trained without any relation-specific input. We also show that the combination of unlexicalized, open extraction in O-CRF and lexicalized, supervised extraction in R1-CRF improves precision and F-measure compared to a standalone RE system.

### 5.1 Open Extraction

This section contrasts the performance of O-CRF with that of O-NB on an Open IE task, and shows that O-CRF achieves both double the recall and increased precision relative to O-NB. For this exper-

iment, we used the set of 500 sentences<sup>3</sup> described in Section 2. Both IE systems were designed and trained prior to the examination of the sample sentences; thus the results on this sentence sample provide a fair measurement of their performance.

While the TEXTRUNNER system was previously found to extract over 7.5 million tuples from a corpus of 9 million Web pages, these experiments are the first to assess its true recall over a known set of relational tuples. As reported in Table 2, O-CRF extracts relational tuples with a precision of 88.3% and a recall of 45.2%. O-CRF achieves a relative gain in F1 of 63.4% over the O-NB model employed by TEXTRUNNER, which obtains a precision of 86.6% and a recall of 23.2%. The recall of O-CRF nearly doubles that of O-NB.

O-CRF is able to extract instances of the four most frequently observed relation types – Verb, Noun+Prep, Verb+Prep and Infinitive. Three of the four remaining types – Modifier, Coordinate<sub>n</sub> and Coordinate<sub>v</sub> – which comprise only 8% of the sample, are not handled due to simplifying assumptions made by both O-CRF and O-NB that tokens indicating a relation occur between entity mentions in the sentence.

### 5.2 O-CRF vs. R1-CRF Extraction

To compare performance of the extractors when a small set of target relationships is known in advance, we used labeled data for four different relations – corporate acquisitions, birthplaces, inventors of products and award winners. The first two datasets were collected from the Web, and made available by Bunescu and Mooney (2007). To augment the size of our corpus, we used the same technique to collect data for two additional relations, and manually labelled positive and negative instances by hand over all collections. For each of the four relations in our collection, we trained R1-CRF from labeled training data, and ran each of R1-CRF and O-CRF over the respective test sets, and compared the precision and recall of all tuples output by each system.

Table 3 shows that from the start, O-CRF achieves a high level of precision – 75.0% – without any

<sup>3</sup>Available at <http://www.cs.washington.edu/research/knowitall/hlt-naac108-data.txt>

Relation	O-CRF		R1-CRF		
	P	R	P	R	Train Ex
Acquisition	75.6	19.5	67.6	69.2	3042
Birthplace	90.6	31.1	92.3	64.4	1853
InventorOf	88.0	17.5	81.3	50.8	682
WonAward	62.5	15.3	73.6	52.8	354
<b>All</b>	<b>75.0</b>	18.4	73.9	<b>58.4</b>	5930

Table 3: Precision (P) and Recall (R) of O-CRF and R1-CRF.

Relation	O-CRF		R1-CRF		
	P	R	P	R	Train Ex
Acquisition	75.6	19.5	67.6	69.2	3042*
Birthplace	90.6	31.1	92.3	53.3	600
InventorOf	88.0	17.5	81.3	50.8	682*
WonAward	62.5	15.3	65.4	61.1	50
<b>All</b>	<b>75.0</b>	18.4	70.17	<b>60.7</b>	>4374

Table 4: For 4 relations, a minimum of 4374 hand-tagged examples is needed for R1-CRF to approximately match the precision of O-CRF for each relation. A “\*” indicates the use of all available training data; in these cases, R1-CRF was unable to match the precision of O-CRF.

relation-specific data. Using labeled training data, the R1-CRF system achieves a slightly lower precision of 73.9%.

Exactly how many training examples per relation does it take R1-CRF to achieve a comparable level of precision? We varied the number of training examples given to R1-CRF, and found that in 3 out of 4 cases it takes hundreds, if not thousands of labeled examples for R1-CRF to achieve acceptable levels of precision. In two cases – acquisitions and inventions – R1-CRF is unable to match the precision of O-CRF, even with many labeled examples. Table 4 summarizes these findings.

Using labeled data, R1-CRF obtains a recall of 58.4%, compared to O-CRF, whose recall is 18.4%. A large number of false negatives on the part of O-CRF can be attributed to its lack of lexical features, which are often crucial when part-of-speech tagging errors are present. For instance, in the sentence, “Yahoo To Acquire Inktomi”, “Acquire” is mistaken for a proper noun, and sufficient evidence of the existence of a relationship is absent. The lexicalized R1-CRF extractor is able to recover from this error; the presence of the word “Acquire” is enough to recog-

Relation	R1-CRF			Hybrid		
	P	R	F1	P	R	F1
Acquisition	67.6	69.2	68.4	76.0	67.5	71.5
Birthplace	93.6	64.4	76.3	96.5	62.2	75.6
InventorOf	81.3	50.8	62.5	87.5	52.5	65.6
WonAward	73.6	52.8	61.5	75.0	50.0	60.0
<b>All</b>	73.9	<b>58.4</b>	65.2	<b>79.2</b>	56.9	<b>66.2</b>

Table 5: A hybrid extractor that uses O-CRF improves precision for all relations, at a small cost to recall.

nize the positive instance, despite the incorrect part-of-speech tag.

Another source of recall issues facing O-CRF is its ability to discover synonyms for a given relation. We found that while RESOLVER improves the relative recall of O-CRF by nearly 50%, O-CRF locates fewer synonyms per relation compared to its lexicalized counterpart. With RESOLVER, O-CRF finds an average of 6.5 synonyms per relation compared to R1-CRF’s 16.25.

In light of our findings, the relative tradeoffs of open versus traditional RE are as follows. Open IE automatically offers a high level of precision without requiring manual labor per relation, at the expense of recall. When relationships in a corpus are not known, or their number is massive, Open IE is essential for RE. When higher levels of recall are desirable for a small set of target relations, traditional RE is more appropriate. However, in this case, one must be willing to undertake the cost of acquiring labeled training data for each relation, either via a computational procedure such as bootstrapped learning or by the use of human annotators.

### 5.3 Hybrid Extraction

In this section, we explore the performance of H-CRF, an ensemble-based extractor that learns to perform RE for a set of known relations based on the individual behaviors of O-CRF and R1-CRF.

As shown in Table 5, the use of O-CRF as part of H-CRF, improves precision from 73.9% to 79.2% with only a slight decrease in recall. Overall, F1 improved from 65.2% to 66.2%.

One disadvantage of a stacking-based hybrid system is that labeled training data is still required. In the future, we would like to explore the development of hybrid systems that leverage Open IE methods,

like O-CRF, to reduce the number of training examples required per relation.

## 6 Related Work

TEXTRUNNER, the first Open IE system, is part of a body of work that reflects a growing interest in avoiding relation-specificity during extraction. Sekine (2006) developed a paradigm for “on-demand information extraction” in order to reduce the amount of effort involved when porting IE systems to new domains. Shinyama and Sekine’s “pre-emptive” IE system (2006) discovers relationships from sets of related news articles.

Until recently, most work in RE has been carried out on a per-relation basis. Typically, RE is framed as a binary classification problem: Given a sentence  $S$  and a relation  $R$ , does  $S$  assert  $R$  between two entities in  $S$ ? Representative approaches include (Zelenko et al., 2003) and (Bunescu and Mooney, 2005), which use support-vector machines fitted with language-oriented kernels to classify pairs of entities. Roth and Yih (2004) also described a classification-based framework in which they jointly learn to identify named entities and relations.

Culotta *et al.* (2006) used a CRF for RE, yet their task differs greatly from open extraction. RE was performed from biographical text in which the topic of each document was known. For every entity found in the document, their goal was to predict what relation, if any, it had relative to the page topic, from a set of given relations. Under these restrictions, RE became an instance of entity labeling, where the label assigned to an entity (*e.g.* *Father*) is its relation to the topic of the article.

Others have also found the stacking framework to yield benefits for IE. Freitag (2000) used linear regression to model the relationship between the confidence of several inductive learning algorithms and the probability that a prediction is correct. Over three different document collections, the combined method yielded improvements over the best individual learner for all but one relation. The efficacy of ensemble-based methods for extraction was further investigated by (Sigletos et al., 2005), who experimented with combining the outputs of a rule-based learner, a Hidden Markov Model and a wrapper-induction algorithm in five different domains. Of a

variety ensemble-based methods, stacking proved to consistently outperform the best base-level system, obtaining more precise results at the cost of somewhat lower recall. (Feldman et al., 2005) demonstrated that a hybrid extractor composed of a statistical and knowledge-based models outperform either in isolation.

## 7 Conclusions and Future Work

Our experiments have demonstrated the promise of relation-independent extraction using the Open IE paradigm. We have shown that binary relationships can be categorized using a compact set of lexico-syntactic patterns, and presented O-CRF, a CRF-based Open IE system that can extract different relationships with a precision of 88.3% and a recall of 45.2%<sup>4</sup>. Open IE is essential when the number of relationships of interest is massive or unknown.

Traditional IE is more appropriate for targeted extraction when the number of relations of interest is small and one is willing to incur the cost of acquiring labeled training data. Compared to traditional IE, the recall of our Open IE system is admittedly lower. However, in a targeted extraction scenario, Open IE can still be used to reduce the number of hand-labeled examples. As Table 4 shows, numerous hand-labeled examples (ranging from 50 for one relation to over 3,000 for another) are necessary to match the precision of O-CRF.

In the future, O-CRF’s recall may be improved by enhancements to its ability to locate the various ways in which a given relation is expressed. We also plan to explore the capacity of Open IE to automatically provide labeled training data, when traditional relation extraction is a more appropriate choice.

## Acknowledgments

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, ONR grant N00014-08-1-0431 as well as gifts from Google, and carried out at the University of Washington’s Turing Center. Doug Downey, Stephen Soderland and Dan Weld provided helpful comments on previous drafts.

<sup>4</sup>The TEXTRUNNER Open IE system now indexes extractions found by O-CRF from millions of Web pages, and is located at <http://www.cs.washington.edu/research/textrunner>

## References

- E. Agichtein and L. Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Procs. of the Fifth ACM International Conference on Digital Libraries*.
- M. Banko, M. Cararella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *Procs. of IJCAI*.
- S. Brin. 1998. Extracting Patterns and Relations from the World Wide Web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*, pages 172–183, Valencia, Spain.
- R. Bunescu and R. Mooney. 2005. Subsequence kernels for relation extraction. In *In Procs. of Neural Information Processing Systems*.
- R. Bunescu and R. Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proc. of ACL*.
- A. Culotta and A. McCallum. 2004. Confidence estimation for information extraction. In *Procs of HLT/NAACL*.
- A. Culotta, A. McCallum, and J. Betz. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Procs of HLT/NAACL*, pages 296–303.
- P. Domingos. 1996. Unifying instance-based and rule-based induction. *Machine Learning*, 24(2):141–168.
- 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. Feldman, B. Rosenfeld, and M. Fresko. 2005. Teg - a hybrid approach to information extraction. *Knowledge and Information Systems*, 9(1):1–18.
- D. Freitag. 2000. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2-3):169–202.
- R. Girju, A. Badulescu, and D. Moldovan. 2006. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1).
- M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Procs. of the 14th International Conference on Computational Linguistics*, pages 539–545.
- D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.
- J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Procs. of ICML*.
- A. McCallum. 2002. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>.
- A. E. Monge and C. P. Elkan. 1996. The field matching problem: Algorithms and applications. In *Procs. of KDD*.
- E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-level Boot-strapping. In *Procs. of AAAI-99*, pages 1044–1049.
- D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Procs. of CoNLL*.
- S. Sekine. 2006. On-demand information extraction. In *Proc. of COLING*.
- Y. Shinyama and S. Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proc. of the HLT-NAACL*.
- G. Sigletos, G. Paliouras, C. D. Spyropoulos, and M. Hatzopoulos. 2005. Combining information extraction systems using voting and stacked generalization. *Journal of Machine Learning Research*, 6:1751,1782.
- R. Snow, D. Jurafsky, and A. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*. MIT Press.
- K.M. Ting and I. H. Witten. 1999. Issues in stacked generalization. *Artificial Intelligence Research*, 10:271–289.
- D. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–260.
- A. Yates and O. Etzioni. 2007. Unsupervised resolution of objects and relations on the web. In *Procs of NAACL/HLT*.
- D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *JMLR*, 3:1083–1106.
- B. Zenko and S. Dzeroski. 2002. Stacking with an extended set of meta-level attributes and mlr. In *Proc. of ECML*.



# PDT 2.0 Requirements on a Query Language

**Jiří Mírovský**

Institute of Formal and Applied Linguistics  
Charles University in Prague  
Malostranské nám. 25, 118 00 Prague 1, Czech Republic  
mirovsky@ufal.mff.cuni.cz

## Abstract

Linguistically annotated treebanks play an essential part in the modern computational linguistics. The more complex the treebanks become, the more sophisticated tools are required for using them, namely for searching in the data. We study linguistic phenomena annotated in the Prague Dependency Treebank 2.0 and create a list of requirements these phenomena set on a search tool, especially on its query language.

## 1 Introduction

Searching in a linguistically annotated treebank is a principal task in the modern computational linguistics. A search tool helps extract useful information from the treebank, in order to study the language, the annotation system or even to search for errors in the annotation.

The more complex the treebank is, the more sophisticated the search tool and its query language needs to be. The Prague Dependency Treebank 2.0 (Hajič et al. 2006) is one of the most advanced manually annotated treebanks. We study mainly the tectogrammatical layer of the Prague Dependency Treebank 2.0 (PDT 2.0), which is by far the most advanced and complex layer in the treebank, and show what requirements on a query language the annotated linguistic phenomena bring. We also add requirements set by lower layers of annotation.

In *section 1* (after this introduction) we mention related works on search languages for various

types of corpora. Afterwards, we very shortly introduce PDT 2.0, just to give a general picture of the principles and complexion of the annotation scheme.

In *section 2* we study the annotation manual for the tectogrammatical layer of PDT 2.0 (t-manual, Mikulová et al. 2006) and collect linguistic phenomena that bring special requirements on the query language. We also study lower layers of annotation and add their requirements.

In *section 3* we summarize the requirements in an extensive list of features required from a search language.

We conclude in *section 4*.

### 1.1 Related Work

In Lai, Bird 2004, the authors name seven linguistic queries they consider important representatives for checking a sufficiency of a query language power. They study several query tools and their query languages and compare them on the basis of their abilities to express these seven queries. In Bird et al. 2005, the authors use a revised set of seven key linguistic queries as a basis for forming a list of three expressive features important for linguistic queries. The features are: immediate precedence, subtree scoping and edge alignment. In Bird et al. 2006, another set of seven linguistic queries is used to show a necessity to enhance XPath (a standard query language for XML, Clark, DeRose 1999) to support linguistic queries.

Cassidy 2002 studies adequacy of XQuery (a search language based on XPath, Boag et al. 1999) for searching in hierarchically annotated data. Re-

quirements on a query language for annotation graphs used in speech recognition is also presented in Bird et al. 2000. A description of linguistic phenomena annotated in the Tiger Treebank, along with an introduction to a search tool TigerSearch, developed especially for this treebank, is given in Brants et al. 2002, nevertheless without a systematic study of the required features.

Laura Kallmeyer (Kallmeyer 2000) studies requirements on a query language based on two examples of complex linguistic phenomena taken from the NEGRA corpus and the Penn Treebank, respectively.

To handle alignment information, Merz and Volk 2005 study requirements on a search tool for parallel treebanks.

All the work mentioned above can be used as an ample source of inspiration, though it cannot be applied directly to PDT 2.0. A thorough study of the PDT 2.0 annotation is needed to form conclusions about requirements on a search tool for this dependency tree-based corpus, consisting of several layers of annotation and having an extremely complex annotation scheme, which we shortly describe in the next subsection.

## 1.2 The Prague Dependency Treebank 2.0

The Prague Dependency Treebank 2.0 is a manually annotated corpus of Czech. The texts are annotated on three layers – morphological, analytical and tectogrammatical.

On the morphological layer, each token of every sentence is annotated with a lemma (attribute `m/lemma`), keeping the base form of the token, and a tag (attribute `m/tag`), which keeps its morphological information.

The analytical layer roughly corresponds to the surface syntax of the sentence; the annotation is a single-rooted dependency tree with labeled nodes. Attribute `a/afun` describes the type of dependency between a dependent node and its governor. The order of the nodes from left to right corresponds exactly to the surface order of tokens in the sentence (attribute `a/ord`).

The tectogrammatical layer captures the linguistic meaning of the sentence in its context. Again, the annotation is a dependency tree with labeled nodes (Hajičová 1998). The correspondence of the nodes to the lower layers is often not 1:1 (Mírovský 2006).

Attribute `functor` describes the dependency between a dependent node and its governor. A tectogrammatical lemma (attribute `t_lemma`) is assigned to every node. 16 grammemes (prefixed `gram`) keep additional annotation (e.g. `gram/verbmod` for verbal modality).

Topic and focus (Hajičová et al. 1998) are marked (attribute `tfa`), together with so-called deep word order reflected by the order of nodes in the annotation (attribute `deepord`).

Coreference relations between nodes of certain category types are captured. Each node has a unique identifier (attribute `id`). Attributes `coref_text.rf` and `coref_gram.rf` contain ids of coreferential nodes of the respective types.

## 2 Phenomena and Requirements

We make a list of linguistic phenomena that are annotated in PDT 2.0 and that determine the necessary features of a query language.

Our work is focused on two structured layers of PDT 2.0 – the analytical layer and the tectogrammatical layer. For using the morphological layer exclusively and directly, a very good search tool Manatee/Bonito (Rychlý 2000) can be used. We intend to access the morphological information only from the higher layers, not directly. Since there is relation 1:1 among nodes on the analytical layer (but for the technical root) and tokens on the morphological layer, the morphological information can be easily merged into the analytical layer – the nodes only get additional attributes.

The tectogrammatical layer is by far the most complex layer in PDT 2.0, therefore we start our analysis with a study of the annotation manual for the tectogrammatical layer (`t-manual`, Mikulová et al. 2006) and focus also on the requirements on accessing lower layers with non-1:1 relation. Afterwards, we add some requirements on a query language set by the annotation of the lower layers – the analytical layer and the morphological layer.

During the studies, we have to keep in mind that we do not only want to search for a phenomenon, but also need to study it, which can be a much more complex task. Therefore, it is not sufficient e.g. to find a predicative complement, which is a trivial task, since attribute `functor` of the complement is set to value `COMPL`. In this particular example, we also need to be able to specify in the

query properties of the node the second dependency of the complement goes to, e.g. that it is an Actor.

A summary of the required features on a query language is given in the subsequent section.

## 2.1 The Tectogrammatical Layer

First, we focus on linguistic phenomena annotated on the tectogrammatical layer. T-manual has more than one thousand pages. Most of the manual describes the annotation of simple phenomena that only require a single-node query or a very simple structured query. We mostly focus on those phenomena that bring a special requirement on the query language.

### 2.1.1 Basic Principles

The basic unit of annotation on the tectogrammatical layer of PDT 2.0 is a sentence.

The representation of the tectogrammatical annotation of a sentence is a rooted dependency tree. It consists of a set of nodes and a set of edges. One of the nodes is marked as a root. Each node is a complex unit consisting of a set of pairs attribute-value (t-manual, page 1). The edges express dependency relations between nodes. The edges do not have their own attributes; attributes that logically belong to edges (e.g. type of dependency) are represented as node-attributes (t-manual, page 2).

It implies the first and most basic requirement on the query language: one result of the search is one sentence along with the tree belonging to it. Also, the query language should be able to express node evaluation and tree dependency among nodes in the most direct way.

### 2.1.2 Valency

Valency of semantic verbs, valency of semantic verbal nouns, valency of semantic nouns that represent the nominal part of a complex predicate and valency of some semantic adverbs are annotated fully in the trees (t-manual, pages 162-3). Since the valency of verbs is the most complete in the annotation and since the requirements on searching for valency frames of nouns are the same as of verbs, we will (for the sake of simplicity in expressions) focus on the verbs only. Every verb meaning is assigned a valency frame. Verbs usually have more than one meaning; each is assigned a separate va-

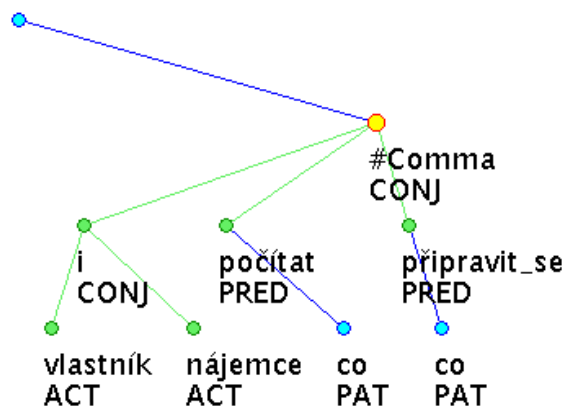
lency frame. Every verb has as many valency frames as it has meanings (t-manual, page 105).

Therefore, the query language has to be able to distinguish valency frames and search for each one of them, at least as long as the valency frames differ in their members and not only in their index. (Two or more identical valency frames may represent different verb meanings (t-manual, page 105).) The required features include a presence of a son, its non-presence, as well as controlling number of sons of a node.

### 2.1.3 Coordination and Apposition

Tree dependency is not always linguistic dependency (t-manual, page 9). Coordination and apposition are examples of such a phenomenon (t-manual, page 282). If a Predicate governs two coordinated Actors, these Actors technically depend on a coordinating node and this coordinating node depends on the Predicate. the query language should be able to skip such a coordinating node. In general, there should be a possibility to skip any type of node.

Skipping a given type of node helps but is not sufficient. The coordinated structure can be more complex, for example the Predicate itself can be coordinated too. Then, the Actors do not even belong to the subtree of any of the Predicates. In the following example, the two Predicates (PRED) are coordinated with conjunction (CONJ), as well as the two Actors (ACT). The linguistic dependencies go from each of the Actors to each of the Predicates but the tree dependencies are quite different:



In Czech: *S čím mohou vlastníci i nájemci počítat, na co by se měli připravit?*

In English: *What can owners and tenants expect, what they should get ready for?*

The query language should therefore be able to express the linguistic dependency directly. The information about the linguistic dependency is annotated in the treebank by the means of references, as well as many other phenomena (see below).

### 2.1.4 Idioms (Phrasemes) etc.

Idioms/phrasemes (idiomatic/phraseologic constructions) are combinations of two or more words with a fixed lexical content, which together constitute one lexical unit with a metaphorical meaning (which cannot be decomposed into meanings of its parts) (t-manual, page 308). Only expressions which are represented by at least two auto-semantic nodes in the tectogrammatical tree are captured as idioms (functor DPHR). One-node (one-auto-semantic-word) idioms are not represented as idioms in the tree. For example, in the combination “chlapec k pohledání” (“a boy to look for”), the prepositional phrase gets functor RSTR, and it is not indicated that it is an idiom.

Secondary prepositions are another example of a linguistic phenomenon that can be easily recognized in the surface form of the sentence but is difficult to find in the tectogrammatical tree.

Therefore, the query language should offer a basic searching in the linear form of the sentence, to allow searching for any idiom or phraseme, regardless of the way it is or is not captured in the tectogrammatical tree. It can even help in a situation when the user does not know how a certain linguistic phenomenon is annotated on the tectogrammatical layer.

### 2.1.5 Complex Predicates

A complex predicate is a multi-word predicate consisting of a semantically empty verb which expresses the grammatical meanings in a sentence, and a noun (frequently denoting an event or a state of affairs) which carries the main lexical meaning of the entire phrase (t-manual, page 345). Searching for a complex predicate is a simple task and does not bring new requirements on the query language. It is valency of complex predicates that requires our attention, especially dual function of a valency modification. The nominal and verbal components of the complex predicate are assigned the appropriate valency frame from the valency lexicon. By means of newly established nodes with `t_lemma` substitutes, those valency modification

positions not present at surface layer are filled. There are problematic cases where the expressed valency modification occurs in the same form in the valency frames of both components of the complex predicate (t-manual, page 362).

To study these special cases of valency, the query language has to offer a possibility to define that a valency member of the verbal part of a complex predicate is at the same time a valency member of the nominal part of the complex predicate, possibly with a different function. The identity of valency members is annotated again by the means of references, which is explained later.

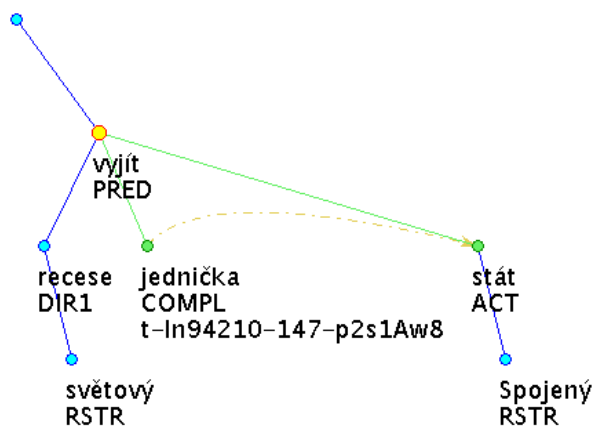
### 2.1.6 Predicative Complement (Dual Dependency)

On the tectogrammatical layer, also cases of the so-called predicative complement are represented. The predicative complement is a non-obligatory free modification (adjunct) which has a dual semantic dependency relation. It simultaneously modifies a noun and a verb (which can be nominalized).

These two dependency relations are represented by different means (t-manual, page 376):

- the dependency on a verb is represented by means of an edge (which means it is represented in the same way like other modifications),
- the dependency on a (semantic) noun is represented by means of attribute `comp.pl.rf`, the value of which is the identifier of the modified noun.

In the following example, the predicative complement (COMPL) has one dependency on a verb (PRED) and another (dual) dependency on a noun (ACT):



In Czech: *Ze světové **recese** vyšly jako jednička Spojené státy.*

In English: *The United States **emerged** from the world **recession** as number one.*

The second form of dependency, represented once again with references (still see below), has to be expressible in the query language.

### 2.1.7 Coreferences

Two types of coreferences are annotated on the tectogrammatical layer:

- grammatical coreference
- textual coreference

The current way of representing coreference uses references (t-manual, page 996).

Let us finally explain what references are. References make use of the fact that every node of every tree has an identifier (the value of attribute `id`), which is unique within PDT 2.0. If coreference, dual dependency, or valency member identity is a link between two nodes (one node referring to another), it is enough to specify the identifier of the referred node in the appropriate attribute of the referring node. Reference types are distinguished by different referring attributes. Individual reference subtypes can be further distinguished by the value of another attribute.

The essential point in references (for the query language) is that at the time of forming a query, the value of the reference is unknown. For example, in the case of dual dependency of predicative complement, we know that the value of attribute `comp1.rf` of the complement must be the same as the value of attribute `id` of the governing noun, but the value itself differs tree from tree and therefore is unknown at the time of creating the query. The query language has to offer a possibility to bind these unknown values.

### 2.1.8 Topic-Focus Articulation

On the tectogrammatical layer, also the topic-focus articulation (TFA) is annotated. TFA annotation comprises two phenomena:

- contextual boundness, which is represented by values of attribute `tfa` for each node of the tectogrammatical tree.
- communicative dynamism, which is represented by the underlying order of nodes.

Annotated trees therefore contain two types of information - on the one hand the value of contextual boundness of a node and its relative ordering with respect to its brother nodes reflects its function within the topic-focus articulation of the sentence, on the other hand the set of all the TFA values in the tree and the relative ordering of subtrees reflect the overall functional perspective of the sentence, and thus enable to distinguish in the sentence the complex categories of topic and focus (however, these are not annotated explicitly) (t-manual, page 1118).

While contextual boundness does not bring any new requirement on the query language, communicative dynamism requires that the relative order of nodes in the tree from left to right can be expressed. The order of nodes is controlled by attribute `deepord`, which contains a non-negative real (usually natural) number that sets the order of the nodes from left to right. Therefore, we will again need to refer to a value of an attribute of another node but this time with relation other than “equal to”.

#### 2.1.8.1 Focus Proper

Focus proper is the most dynamic and communicatively significant contextually non-bound part of the sentence. Focus proper is placed on the rightmost path leading from the effective root of the tectogrammatical tree, even though it is at a different position in the surface structure. The node representing this expression will be placed rightmost in the tectogrammatical tree. If the focus proper is constituted by an expression represented as the effective root of the tectogrammatical tree (i.e. the governing predicate is the focus proper), there is no right path leading from the effective root (t-manual, page 1129).

#### 2.1.8.2 Quasi-Focus

Quasi-focus is constituted by (both contrastive and non-contrastive) contextually bound expressions, on which the focus proper is dependent. The focus proper can immediately depend on the quasi-focus, or it can be a more deeply embedded expression.

In the underlying word order, nodes representing the quasi-focus, although they are contextually bound, are placed to the right from their governing node. Nodes representing the quasi-focus are therefore contextually bound nodes on the rightmost

path in the tectogrammatical tree (t-manual, page 1130).

The ability of the query language to distinguish the rightmost node in the tree and the rightmost path leading from a node is therefore necessary.

### 2.1.8.3 Rhematizers

Rhematizers are expressions whose function is to signal the topic-focus articulation categories in the sentence, namely the communicatively most important categories - the focus and contrastive topic.

The position of rhematizers in the surface word order is quite loose, however they almost always stand right before the expressions they rhematize, i.e. the expressions whose being in the focus or contrastive topic they signal (t-manual, pages 1165-6).

The guidelines for positioning rhematizers in tectogrammatical trees are simple (t-manual, page 1171):

- a rhematizer (i.e. the node representing the rhematizer) is placed as the closest left brother (in the underlying word order) of the first node of the expression that is in its scope.
- if the scope of a rhematizer includes the governing predicate, the rhematizer is placed as the closest left son of the node representing the governing predicate.
- if a rhematizer constitutes the focus proper, it is placed according to the guidelines for the position of the focus proper - i.e. on the rightmost path leading from the effective root of the tectogrammatical tree.

Rhematizers therefore bring a further requirement on the query language – an ability to control the distance between nodes (in the terms of deep word order); at the very least, the query language has to distinguish an immediate brother and relative horizontal position of nodes.

### 2.1.8.4 (Non-)Projectivity

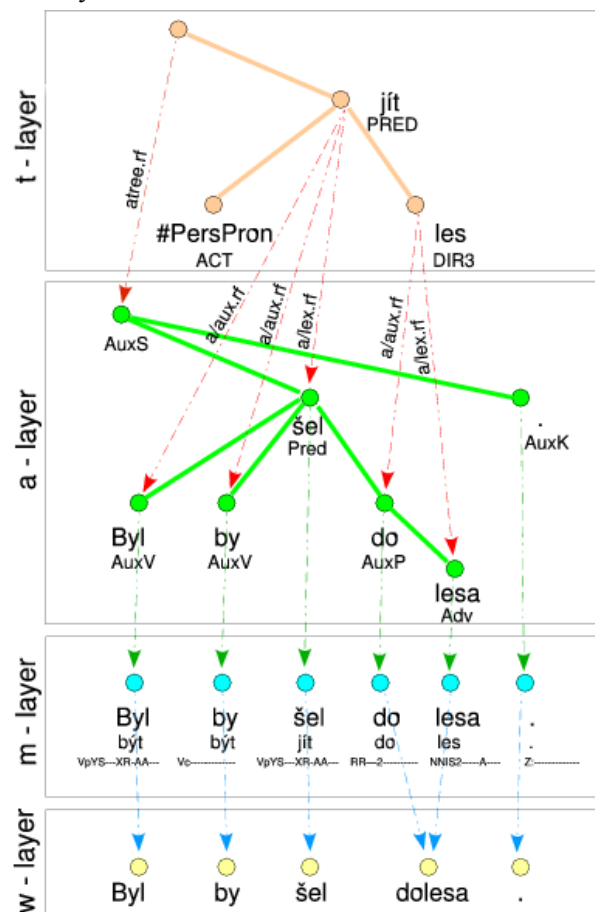
Projectivity of a tree is defined as follows: if two nodes B and C are connected by an edge and C is to the left from B, then all nodes to the right from B and to the left from C are connected with the root via a path that passes through at least one of the nodes B or C. In short: between a father and its son there can only be direct or indirect sons of the father (t-manual, page 1135).

The relative position of a node (node A) and an edge (nodes B, C) that together cause a non-projectivity forms four different configurations: (“B is on the left from C” or “B is on the right from C”) x (“A is on the path from B to the root” or “it is not”). Each of the configurations can be searched for using properties of the language that have been required so far by other linguistic phenomena. Four different queries search for four different configurations.

To be able to search for all configurations in one query, the query language should be able to combine several queries into one multi-query. We do not require that a general logical expression can be set above the single queries. We only require a general OR combination of the single queries.

### 2.1.9 Accessing Lower Layers

Studies of many linguistic phenomena require a multilayer access.



In Czech: *Byl by šel do lesa.*

In English (lit.): *He would have gone to the forest.*

For example, the query “find an example of Patient that is more dynamic than its governing Predicate (with greater `deepord`) but on the surface layer is on the left side from the Predicate” requires information both from the tectogrammatical layer and the analytical layer.

The picture above is taken from PDT 2.0 guide and shows the typical relation among layers of annotation for the sentence (the lowest w-layer is a technical layer containing only the tokenized original data).

The information from the lower layers can be easily compressed into the analytical layer, since there is relation 1:1 among the layers (with some rare exceptions like misprints in the w-layer). The situation between the tectogrammatical layer and the analytical layer is much more complex. Several nodes from the analytical layer may be (and often are) represented by one node on the tectogrammatical layer and new nodes without an analytical counterpart may appear on the tectogrammatical layer. It is necessary that the query language addresses this issue and allows access to the information from the lower layers.

## 2.2 The Analytical and Morphological Layer

The analytical layer is much less complex than the tectogrammatical layer. The basic principles are the same – the representation of the structure of a sentence is rendered in the form of a tree – a connected acyclic directed graph in which no more than one edge leads into a node, and whose nodes are labeled with complex symbols (sets of attributes). The edges are not labeled (in the technical sense). The information logically belonging to an edge is represented in attributes of the depending node. One node is marked as a root.

Here, we focus on linguistic phenomena annotated on the analytical and morphological layer that bring a new requirement on the query language (that has not been set in the studies of the tectogrammatical layer).

### 2.2.1 Morphological Tags

In PDT 2.0, morphological tags are positional. They consist of 15 characters, each representing a certain morphological category, e.g. the first position represents part of speech, the third position represents gender, the fourth position represents number, the fifth position represents case.

The query language has to offer a possibility to specify a part of the tag and leave the rest unspecified. It has to be able to set such conditions on the tag like “this is a noun”, or “this is a plural in fourth case”. Some conditions might include negation or enumeration, like “this is an adjective that is not in fourth case”, or “this is a noun either in third or fourth case”. This is best done with some sort of wild cards. The latter two examples suggest that such a strong tool like regular expressions may be needed.

### 2.2.2 Agreement

There are several cases of agreement in Czech language, like agreement in case, number and gender in attributive adjective phrase, agreement in gender and number between predicate and subject (though it may be complex), or agreement in case in apposition.

To study agreement, the query language has to allow to make a reference to only a part of value of attribute of another node, e.g. to the fifth position of the morphological tag for case.

### 2.2.3 Word Order

Word order is a linguistic phenomenon widely studied on the analytical layer, because it offers a perfect combination of a word order (the same like in the sentence) and syntactic relations between the words. The same technique like with the deep word order on the tectogrammatical layer can be used here. The order of words (tokens) ~ nodes in the analytical tree is controlled by attribute `ord`. Non-projective constructions are much more often and interesting here than on the tectogrammatical layer. Nevertheless, they appear also on the tectogrammatical layer and their contribution to the requirements on the query language has already been mentioned.

The only new requirement on the query language is an ability to measure the horizontal distance between words, to satisfy linguistic queries like “find trees where a preposition and the head of the noun phrase are at least five words apart”.

## 3 Summary of the Features

Here we summarize what features the query language has to have to suit PDT 2.0. We list the features from the previous section and also add some

obvious requirements that have not been mentioned so far but are very useful generally, regardless of a corpus.

### 3.1 Complex Evaluation of a Node

- multiple attributes evaluation (an ability to set values of several attributes at one node)
- alternative values (e.g. to define that `functor` of a node is either a disjunction or a conjunction)
- alternative nodes (alternative evaluation of the whole set of attributes of a node)
- wild cards (regular expressions) in values of attributes (e.g. `m/tag="N...4.*"` defines that the morphological tag of a node is a noun in accusative, regardless of other morphological categories)
- negation (e.g. to express "this node is not Actor")
- relations less than (`<=`) , greater than (`>=`) (for numerical attributes)

### 3.2 Dependencies Between Nodes (Vertical Relations)

- immediate, transitive dependency (existence, non-existence)
- vertical distance (from root, from one another)
- number of sons (zero for lists)

### 3.3 Horizontal Relations

- precedence, immediate precedence, horizontal distance (all both positive, negative)
- secondary edges, secondary dependencies, coreferences, long-range relations

### 3.4 Other Features

- multiple-tree queries (combined with general OR relation)
- skipping a node of a given type (for skipping simple types of coordination, apposition etc.)
- skipping multiple nodes of a given type (e.g. for recognizing the rightmost path)
- references (for matching values of attributes unknown at the time of creating the query)

- accessing several layers of annotation at the same time with non-1:1 relation (for studying relation between layers)
- searching in the surface form of the sentence

## 4 Conclusion

We have studied the Prague Dependency Treebank 2.0 tectogrammatical annotation manual and listed linguistic phenomena that require a special feature from any query tool for this corpus. We have also added several other requirements from the lower layers of annotation. We have summarized these features, along with general corpus-independent features, in a concise list.

## Acknowledgment

This research was supported by the Grant Agency of the Academy of Sciences of the Czech Republic, project IS-REST (No. 1ET101120413).

## References

- Bird et al. 2000. Towards A Query Language for Annotation Graphs. In: *Proceedings of the Second International Language and Evaluation Conference, Paris, ELRA, 2000*.
- Bird et al. 2005. Extending XPath to Support Linguistic Queries. In: *Proceedings of the Workshop on Programming Language Technologies for XML, California, USA, 2005*.
- Bird et al. 2006. Designing and Evaluating an XPath Dialect for Linguistic Queries. In: *Proceedings of the 22nd International Conference on Data Engineering (ICDE), pp 52-61, Atlanta, USA, 2006*.
- Boag et al. 1999. XQuery 1.0: An XML Query Language. *IW3C Working Draft*, <http://www.w3.org/TR/xpath>, 1999.
- Brants S. et al. 2002. The TIGER Treebank. In: *Proceedings of TLT 2002, Sozopol, Bulgaria, 2002*.
- Cassidy S. 2002. XQuery as an Annotation Query Language: a Use Case Analysis. In: *Proceedings of the Third International Conference on Language Resources and Evaluation, Canary Islands, Spain, 2002*
- Clark J., DeRose S. 1999. XML Path Language (XPath). <http://www.w3.org/TR/xpath>, 1999.
- Hajič J. et al. 2006. Prague Dependency Treebank 2.0. CD-ROM LDC2006T01, LDC, Philadelphia, 2006.



- Hajičová E. 1998. Prague Dependency Treebank: From analytic to tectogrammatical annotations. In: *Proceedings of 2nd TST, Brno, Springer-Verlag Berlin Heidelberg New York, 1998, pp. 45-50.*
- Hajičová E., Partee B., Sgall P. 1998. Topic-Focus Articulation, Tripartite Structures and Semantic Content. *Dordrecht, Amsterdam, Kluwer Academic Publishers, 1998.*
- Havelka J. 2007. Beyond Projectivity: Multilingual Evaluation of Constraints and Measures on Non-Projective Structures. In *Proceedings of ACL 2007, Prague, pp. 608-615.*
- Kallmeyer L. 2000: On the Complexity of Queries for Structurally Annotated Linguistic Data. In *Proceedings of ACIDCA'2000, Corpora and Natural Language Processing, Tunisia, 2000, pp. 105-110.*
- Lai C., Bird S. 2004. Querying and updating treebanks: A critical survey and requirements analysis. In: *Proceedings of the Australasian Language Technology Workshop, Sydney, Australia, 2004*
- Merz Ch., Volk M. 2005. Requirements for a Parallel Treebank Search Tool. In: *Proceedings of GLDV-Conference, Bonn, Germany, 2005.*
- Mikulová et al. 2006. Annotation on the Tectogrammatical Level in the Prague Dependency Treebank (Reference Book). *ÚFAL/CKL Technical Report TR-2006-32, Charles University in Prague, 2006.*
- Mírovský J. 2006. Netgraph: a Tool for Searching in Prague Dependency Treebank 2.0. In *Proceedings of TLT 2006, Prague, pp. 211-222.*
- Rychlý P. 2000. Korpusové manažery a jejich efektivní implementace. *PhD. Thesis, Brno, 2000.*

# Task-oriented Evaluation of Syntactic Parsers and Their Representations

Yusuke Miyao<sup>†</sup> Rune Sætre<sup>†</sup> Kenji Sagae<sup>†</sup> Takuya Matsuzaki<sup>†</sup> Jun'ichi Tsujii<sup>†‡\*</sup>

<sup>†</sup>Department of Computer Science, University of Tokyo, Japan

<sup>‡</sup>School of Computer Science, University of Manchester, UK

<sup>\*</sup>National Center for Text Mining, UK

{yusuke, rune.saetre, sagae, matuzaki, tsujii}@is.s.u-tokyo.ac.jp

## Abstract

This paper presents a comparative evaluation of several state-of-the-art English parsers based on different frameworks. Our approach is to measure the impact of each parser when it is used as a component of an information extraction system that performs protein-protein interaction (PPI) identification in biomedical papers. We evaluate eight parsers (based on dependency parsing, phrase structure parsing, or deep parsing) using five different parse representations. We run a PPI system with several combinations of parser and parse representation, and examine their impact on PPI identification accuracy. Our experiments show that the levels of accuracy obtained with these different parsers are similar, but that accuracy improvements vary when the parsers are re-trained with domain-specific data.

## 1 Introduction

Parsing technologies have improved considerably in the past few years, and high-performance syntactic parsers are no longer limited to PCFG-based frameworks (Charniak, 2000; Klein and Manning, 2003; Charniak and Johnson, 2005; Petrov and Klein, 2007), but also include dependency parsers (McDonald and Pereira, 2006; Nivre and Nilsson, 2005; Sagae and Tsujii, 2007) and deep parsers (Kaplan et al., 2004; Clark and Curran, 2004; Miyao and Tsujii, 2008). However, efforts to perform extensive comparisons of syntactic parsers based on different frameworks have been limited. The most popular method for parser comparison involves the direct measurement of the parser output accuracy in terms of metrics such as bracketing precision and recall, or

dependency accuracy. This assumes the existence of a gold-standard test corpus, such as the Penn Treebank (Marcus et al., 1994). It is difficult to apply this method to compare parsers based on different frameworks, because parse representations are often framework-specific and differ from parser to parser (Ringger et al., 2004). The lack of such comparisons is a serious obstacle for NLP researchers in choosing an appropriate parser for their purposes.

In this paper, we present a comparative evaluation of syntactic parsers and their output representations based on different frameworks: dependency parsing, phrase structure parsing, and deep parsing. Our approach to parser evaluation is to measure accuracy improvement in the task of identifying protein-protein interaction (PPI) information in biomedical papers, by incorporating the output of different parsers as statistical features in a machine learning classifier (Yakushiji et al., 2005; Katrenko and Adriaans, 2006; Erkan et al., 2007; Sætre et al., 2007). PPI identification is a reasonable task for parser evaluation, because it is a typical information extraction (IE) application, and because recent studies have shown the effectiveness of syntactic parsing in this task. Since our evaluation method is applicable to any parser output, and is grounded in a real application, it allows for a fair comparison of syntactic parsers based on different frameworks.

Parser evaluation in PPI extraction also illuminates domain portability. Most state-of-the-art parsers for English were trained with the Wall Street Journal (WSJ) portion of the Penn Treebank, and high accuracy has been reported for WSJ text; however, these parsers rely on lexical information to attain high accuracy, and it has been criticized that these parsers may overfit to WSJ text (Gildea, 2001;

Klein and Manning, 2003). Another issue for discussion is the portability of training methods. When training data in the target domain is available, as is the case with the GENIA Treebank (Kim et al., 2003) for biomedical papers, a parser can be re-trained to adapt to the target domain, and larger accuracy improvements are expected, if the training method is sufficiently general. We will examine these two aspects of domain portability by comparing the original parsers with the retrained parsers.

## 2 Syntactic Parsers and Their Representations

This paper focuses on eight representative parsers that are classified into three parsing frameworks: *dependency parsing*, *phrase structure parsing*, and *deep parsing*. In general, our evaluation methodology can be applied to English parsers based on any framework; however, in this paper, we chose parsers that were originally developed and trained with the Penn Treebank or its variants, since such parsers can be re-trained with GENIA, thus allowing for us to investigate the effect of domain adaptation.

### 2.1 Dependency parsing

Because the shared tasks of CoNLL-2006 and CoNLL-2007 focused on data-driven dependency parsing, it has recently been extensively studied in parsing research. The aim of dependency parsing is to compute a tree structure of a sentence where nodes are words, and edges represent the relations among words. Figure 1 shows a dependency tree for the sentence “IL-8 recognizes and activates CXCR1.” An advantage of dependency parsing is that dependency trees are a reasonable approximation of the semantics of sentences, and are readily usable in NLP applications. Furthermore, the efficiency of popular approaches to dependency parsing compare favorably with those of phrase structure parsing or deep parsing. While a number of approaches have been proposed for dependency parsing, this paper focuses on two typical methods.

**MST** McDonald and Pereira (2006)’s dependency parser,<sup>1</sup> based on the Eisner algorithm for projective dependency parsing (Eisner, 1996) with the second-order factorization.

<sup>1</sup><http://sourceforge.net/projects/mstparser>

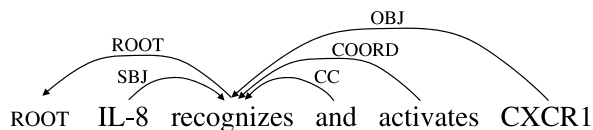


Figure 1: CoNLL-X dependency tree

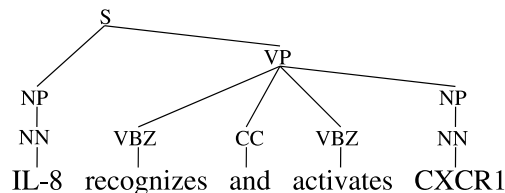


Figure 2: Penn Treebank-style phrase structure tree

**KSDEP** Sagae and Tsujii (2007)’s dependency parser,<sup>2</sup> based on a probabilistic shift-reduce algorithm extended by the pseudo-projective parsing technique (Nivre and Nilsson, 2005).

### 2.2 Phrase structure parsing

Owing largely to the Penn Treebank, the mainstream of data-driven parsing research has been dedicated to the phrase structure parsing. These parsers output Penn Treebank-style phrase structure trees, although function tags and empty categories are stripped off (Figure 2). While most of the state-of-the-art parsers are based on probabilistic CFGs, the parameterization of the probabilistic model of each parser varies. In this work, we chose the following four parsers.

**NO-RERANK** Charniak (2000)’s parser, based on a lexicalized PCFG model of phrase structure trees.<sup>3</sup> The probabilities of CFG rules are parameterized on carefully hand-tuned extensive information such as lexical heads and symbols of ancestor/sibling nodes.

**RERANK** Charniak and Johnson (2005)’s reranking parser. The reranker of this parser receives  $n$ -best<sup>4</sup> parse results from NO-RERANK, and selects the most likely result by using a maximum entropy model with manually engineered features.

**BERKELEY** Berkeley’s parser (Petrov and Klein, 2007).<sup>5</sup> The parameterization of this parser is op-

<sup>2</sup><http://www.cs.cmu.edu/~sagae/parser/>

<sup>3</sup><http://bllip.cs.brown.edu/resources.shtml>

<sup>4</sup>We set  $n = 50$  in this paper.

<sup>5</sup><http://nlp.cs.berkeley.edu/Main.html#Parsing>

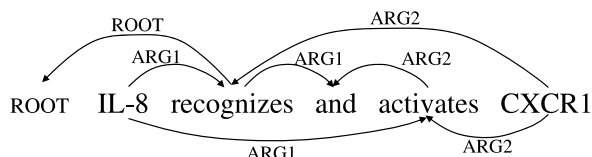


Figure 3: Predicate argument structure

timized automatically by assigning latent variables to each nonterminal node and estimating the parameters of the latent variables by the EM algorithm (Matsuzaki et al., 2005).

**STANFORD** Stanford’s unlexicalized parser (Klein and Manning, 2003).<sup>6</sup> Unlike NO-RERANK, probabilities are not parameterized on lexical heads.

### 2.3 Deep parsing

Recent research developments have allowed for efficient and robust deep parsing of real-world texts (Kaplan et al., 2004; Clark and Curran, 2004; Miyao and Tsujii, 2008). While deep parsers compute theory-specific syntactic/semantic structures, predicate argument structures (PAS) are often used in parser evaluation and applications. PAS is a graph structure that represents syntactic/semantic relations among words (Figure 3). The concept is therefore similar to CoNLL dependencies, though PAS expresses deeper relations, and may include reentrant structures. In this work, we chose the two versions of the Enju parser (Miyao and Tsujii, 2008).

**ENJU** The HPSG parser that consists of an HPSG grammar extracted from the Penn Treebank, and a maximum entropy model trained with an HPSG treebank derived from the Penn Treebank.<sup>7</sup>

**ENJU-GENIA** The HPSG parser adapted to biomedical texts, by the method of Hara et al. (2007). Because this parser is trained with both WSJ and GENIA, we compare it parsers that are retrained with GENIA (see section 3.3).

## 3 Evaluation Methodology

In our approach to parser evaluation, we measure the accuracy of a PPI extraction system, in which

<sup>6</sup><http://nlp.stanford.edu/software/lex-parser.shtml>

<sup>7</sup><http://www-tsujii.is.s.u-tokyo.ac.jp/enju/>

This study demonstrates that **IL-8** recognizes and activates **CXCR1**, **CXCR2**, and the **Duffy antigen** by distinct mechanisms.

The molar ratio of serum **retinol-binding protein (RBP)** to **transthyretin (TTR)** is not useful to assess vitamin A status during infection in hospitalised children.

Figure 4: Sentences including protein names

ENTITY1(IL-8)  $\xrightarrow{\text{SBJ}}$  recognizes  $\xleftarrow{\text{OBJ}}$  ENTITY2(CXCR1)

Figure 5: Dependency path

the parser output is embedded as statistical features of a machine learning classifier. We run a classifier with features of every possible combination of a parser and a parse representation, by applying conversions between representations when necessary. We also measure the accuracy improvements obtained by parser retraining with GENIA, to examine the domain portability, and to evaluate the effectiveness of domain adaptation.

### 3.1 PPI extraction

PPI extraction is an NLP task to identify protein pairs that are mentioned as interacting in biomedical papers. Because the number of biomedical papers is growing rapidly, it is impossible for biomedical researchers to read all papers relevant to their research; thus, there is an emerging need for reliable IE technologies, such as PPI identification.

Figure 4 shows two sentences that include protein names: the former sentence mentions a protein interaction, while the latter does not. Given a protein pair, PPI extraction is a task of binary classification; for example,  $\langle \text{IL-8}, \text{CXCR1} \rangle$  is a positive example, and  $\langle \text{RBP}, \text{TTR} \rangle$  is a negative example. Recent studies on PPI extraction demonstrated that dependency relations between target proteins are effective features for machine learning classifiers (Kartrenko and Adriaans, 2006; Erkan et al., 2007; Sætre et al., 2007). For the protein pair **IL-8** and **CXCR1** in Figure 4, a dependency parser outputs a dependency tree shown in Figure 1. From this dependency tree, we can extract a dependency path shown in Figure 5, which appears to be a strong clue in knowing that these proteins are mentioned as interacting.

```
(dep_path (SBJ (ENTITY1 recognizes))
  (rOBJ (recognizes ENTITY2)))
```

Figure 6: Tree representation of a dependency path

We follow the PPI extraction method of Sætne et al. (2007), which is based on SVMs with SubSet Tree Kernels (Collins and Duffy, 2002; Moschitti, 2006), while using different parsers and parse representations. Two types of features are incorporated in the classifier. The first is bag-of-words features, which are regarded as a strong baseline for IE systems. Lemmas of words before, between and after the pair of target proteins are included, and the linear kernel is used for these features. These features are commonly included in all of the models. Filtering by a stop-word list is not applied because this setting made the scores higher than Sætne et al. (2007)’s setting. The other type of feature is syntactic features. For dependency-based parse representations, a dependency path is encoded as a flat tree as depicted in Figure 6 (prefix “r” denotes reverse relations). Because a tree kernel measures the similarity of trees by counting common subtrees, it is expected that the system finds effective subsequences of dependency paths. For the PTB representation, we directly encode phrase structure trees.

### 3.2 Conversion of parse representations

It is widely believed that the choice of representation format for parser output may greatly affect the performance of applications, although this has not been extensively investigated. We should therefore evaluate the parser performance in multiple parse representations. In this paper, we create multiple parse representations by converting each parser’s default output into other representations when possible. This experiment can also be considered to be a comparative evaluation of parse representations, thus providing an indication for selecting an appropriate parse representation for similar IE tasks.

Figure 7 shows our scheme for representation conversion. This paper focuses on five representations as described below.

**CoNLL** The dependency tree format used in the 2006 and 2007 CoNLL shared tasks on dependency parsing. This is a representation format supported by several data-driven dependency parsers. This repre-

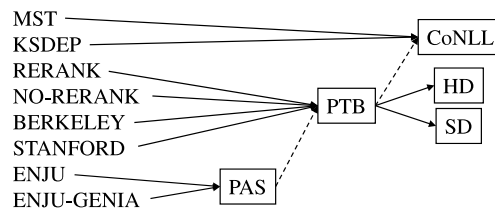


Figure 7: Conversion of parse representations

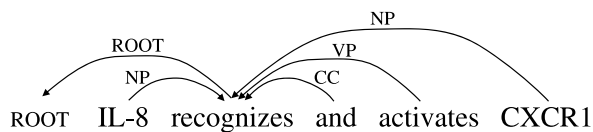


Figure 8: Head dependencies

sation is also obtained from Penn Treebank-style trees by applying constituent-to-dependency conversion<sup>8</sup> (Johansson and Nugues, 2007). It should be noted, however, that this conversion cannot work perfectly with automatic parsing, because the conversion program relies on function tags and empty categories of the original Penn Treebank.

**PTB** Penn Treebank-style phrase structure trees without function tags and empty nodes. This is the default output format for phrase structure parsers. We also create this representation by converting ENJU’s output by tree structure matching, although this conversion is not perfect because forms of PTB and ENJU’s output are not necessarily compatible.

**HD** Dependency trees of syntactic heads (Figure 8). This representation is obtained by converting PTB trees. We first determine lexical heads of nonterminal nodes by using Bikel’s implementation of Collins’ head detection algorithm<sup>9</sup> (Bikel, 2004; Collins, 1997). We then convert lexicalized trees into dependencies between lexical heads.

**SD** The Stanford dependency format (Figure 9). This format was originally proposed for extracting dependency relations useful for practical applications (de Marneffe et al., 2006). A program to convert PTB is attached to the Stanford parser. Although the concept looks similar to CoNLL, this representa-

<sup>8</sup><http://nlp.cs.lth.se/pennconverter/>

<sup>9</sup><http://www.cis.upenn.edu/~dbikel/software.html>

html

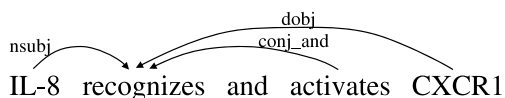


Figure 9: Stanford dependencies

tion does not necessarily form a tree structure, and is designed to express more fine-grained relations such as apposition. Research groups for biomedical NLP recently adopted this representation for corpus annotation (Pyysalo et al., 2007a) and parser evaluation (Clegg and Shepherd, 2007; Pyysalo et al., 2007b).

**PAS** Predicate-argument structures. This is the default output format for ENJU and ENJU-GENIA.

Although only CoNLL is available for dependency parsers, we can create four representations for the phrase structure parsers, and five for the deep parsers. Dotted arrows in Figure 7 indicate imperfect conversion, in which the conversion inherently introduces errors, and may decrease the accuracy. We should therefore take caution when comparing the results obtained by imperfect conversion. We also measure the accuracy obtained by the ensemble of two parsers/representations. This experiment indicates the differences and overlaps of information conveyed by a parser or a parse representation.

### 3.3 Domain portability and parser retraining

Since the domain of our target text is different from WSJ, our experiments also highlight the domain portability of parsers. We run two versions of each parser in order to investigate the two types of domain portability. First, we run the original parsers trained with WSJ<sup>10</sup> (39832 sentences). The results in this setting indicate the domain portability of the original parsers. Next, we run parsers re-trained with GENIA<sup>11</sup> (8127 sentences), which is a Penn Treebank-style treebank of biomedical paper abstracts. Accuracy improvements in this setting indicate the possibility of domain adaptation, and the portability of the training methods of the parsers. Since the parsers listed in Section 2 have programs for the training

<sup>10</sup>Some of the parser packages include parsing models trained with extended data, but we used the models trained with WSJ section 2-21 of the Penn Treebank.

<sup>11</sup>The domains of GENIA and AImed are not exactly the same, because they are collected independently.

with a Penn Treebank-style treebank, we use those programs as-is. Default parameter settings are used for this parser re-training.

In preliminary experiments, we found that dependency parsers attain higher dependency accuracy when trained only with GENIA. We therefore only input GENIA as the training data for the retraining of dependency parsers. For the other parsers, we input the concatenation of WSJ and GENIA for the retraining, while the reranker of RERANK was not re-trained due to its cost. Since the parsers other than NO-RERANK and RERANK require an external POS tagger, a WSJ-trained POS tagger is used with WSJ-trained parsers, and *geniatagger* (Tsuruoka et al., 2005) is used with GENIA-retrained parsers.

## 4 Experiments

### 4.1 Experiment settings

In the following experiments, we used AImed (Bunescu and Mooney, 2004), which is a popular corpus for the evaluation of PPI extraction systems. The corpus consists of 225 biomedical paper abstracts (1970 sentences), which are sentence-split, tokenized, and annotated with proteins and PPIs. We use gold protein annotations given in the corpus. Multi-word protein names are concatenated and treated as single words. The accuracy is measured by abstract-wise 10-fold cross validation and the one-answer-per-occurrence criterion (Giuliano et al., 2006). A threshold for SVMs is moved to adjust the balance of precision and recall, and the maximum f-scores are reported for each setting.

### 4.2 Comparison of accuracy improvements

Tables 1 and 2 show the accuracy obtained by using the output of each parser in each parse representation. The row “baseline” indicates the accuracy obtained with bag-of-words features. Table 3 shows the time for parsing the entire AImed corpus, and Table 4 shows the time required for 10-fold cross validation with GENIA-retrained parsers.

When using the original WSJ-trained parsers (Table 1), all parsers achieved almost the same level of accuracy — a significantly better result than the baseline. To the extent of our knowledge, this is the first result that proves that dependency parsing, phrase structure parsing, and deep parsing perform

	CoNLL	PTB	HD	SD	PAS
baseline	48.2/54.9/51.1				
MST	53.2/56.5/54.6	N/A	N/A	N/A	N/A
KSDEP	49.3/63.0/55.2	N/A	N/A	N/A	N/A
NO-RERANK	50.7/60.9/55.2	45.9/60.5/52.0	50.6/60.9/55.1	49.9/58.2/53.5	N/A
RERANK	53.6/59.2/56.1	47.0/58.9/52.1	48.1/65.8/55.4	50.7/62.7/55.9	N/A
BERKELEY	45.8/67.6/54.5	50.5/57.6/53.7	52.3/58.8/55.1	48.7/62.4/54.5	N/A
STANFORD	50.4/60.6/54.9	50.9/56.1/53.0	50.7/60.7/55.1	51.8/58.1/54.5	N/A
ENJU	52.6/58.0/55.0	48.7/58.8/53.1	57.2/51.9/54.2	52.2/58.1/54.8	48.9/64.1/55.3

Table 1: Accuracy on the PPI task with WSJ-trained parsers (precision/recall/f-score)

	CoNLL	PTB	HD	SD	PAS
baseline	48.2/54.9/51.1				
MST	49.1/65.6/55.9	N/A	N/A	N/A	N/A
KSDEP	51.6/67.5/58.3	N/A	N/A	N/A	N/A
NO-RERANK	53.9/60.3/56.8	51.3/54.9/52.8	53.1/60.2/56.3	54.6/58.1/56.2	N/A
RERANK	52.8/61.5/56.6	48.3/58.0/52.6	52.1/60.3/55.7	53.0/61.1/56.7	N/A
BERKELEY	52.7/60.3/56.0	48.0/59.9/53.1	54.9/54.6/54.6	50.5/63.2/55.9	N/A
STANFORD	49.3/62.8/55.1	44.5/64.7/52.5	49.0/62.0/54.5	54.6/57.5/55.8	N/A
ENJU	54.4/59.7/56.7	48.3/60.6/53.6	56.7/55.6/56.0	54.4/59.3/56.6	52.0/63.8/57.2
ENJU-GENIA	56.4/57.4/56.7	46.5/63.9/53.7	53.4/60.2/56.4	55.2/58.3/56.5	57.5/59.8/58.4

Table 2: Accuracy on the PPI task with GENIA-retrained parsers (precision/recall/f-score)

	WSJ-trained	GENIA-retrained
MST	613	425
KSDEP	136	111
NO-RERANK	2049	1372
RERANK	2806	2125
BERKELEY	1118	1198
STANFORD	1411	1645
ENJU	1447	727
ENJU-GENIA		821

Table 3: Parsing time (sec.)

	CoNLL	PTB	HD	SD	PAS
baseline	424				
MST	809	N/A	N/A	N/A	N/A
KSDEP	864	N/A	N/A	N/A	N/A
NO-RERANK	851	4772	882	795	N/A
RERANK	849	4676	881	778	N/A
BERKELEY	869	4665	895	804	N/A
STANFORD	847	4614	886	799	N/A
ENJU	832	4611	884	789	1005
ENJU-GENIA	874	4624	895	783	1020

Table 4: Evaluation time (sec.)

equally well in a real application. Among these parsers, RERANK performed slightly better than the other parsers, although the difference in the f-score is small, while it requires much higher parsing cost.

When the parsers are retrained with GENIA (Table 2), the accuracy increases significantly, demonstrating that the WSJ-trained parsers are not sufficiently domain-independent, and that domain adaptation is effective. It is an important observation that the improvements by domain adaptation are larger than the differences among the parsers in the previous experiment. Nevertheless, not all parsers had their performance improved upon retraining. Parser

retraining yielded only slight improvements for RERANK, BERKELEY, and STANFORD, while larger improvements were observed for MST, KSDEP, NO-RERANK, and ENJU. Such results indicate the differences in the portability of training methods. A large improvement from ENJU to ENJU-GENIA shows the effectiveness of the specifically designed domain adaptation method, suggesting that the other parsers might also benefit from more sophisticated approaches for domain adaptation.

While the accuracy level of PPI extraction is the similar for the different parsers, parsing speed

		RERANK			ENJU			
		CoNLL	HD	SD	CoNLL	HD	SD	PAS
KSDEP	CoNLL	58.5 (+0.2)	57.1 (-1.2)	58.4 (+0.1)	58.5 (+0.2)	58.0 (-0.3)	59.1 (+0.8)	59.0 (+0.7)
RERANK	CoNLL		56.7 (+0.1)	57.1 (+0.4)	58.3 (+1.6)	57.3 (+0.7)	58.7 (+2.1)	59.5 (+2.3)
	HD			56.8 (+0.1)	57.2 (+0.5)	56.5 (+0.5)	56.8 (+0.2)	57.6 (+0.4)
	SD				58.3 (+1.6)	58.3 (+1.6)	56.9 (+0.2)	58.6 (+1.4)
ENJU	CoNLL					57.0 (+0.3)	57.2 (+0.5)	58.4 (+1.2)
	HD						57.1 (+0.5)	58.1 (+0.9)
	SD							58.3 (+1.1)

Table 5: Results of parser/representation ensemble (f-score)

differs significantly. The dependency parsers are much faster than the other parsers, while the phrase structure parsers are relatively slower, and the deep parsers are in between. It is noteworthy that the dependency parsers achieved comparable accuracy with the other parsers, while they are more efficient.

The experimental results also demonstrate that PTB is significantly worse than the other representations with respect to cost for training/testing and contributions to accuracy improvements. The conversion from PTB to dependency-based representations is therefore desirable for this task, although it is possible that better results might be obtained with PTB if a different feature extraction mechanism is used. Dependency-based representations are competitive, while CoNLL seems superior to HD and SD in spite of the imperfect conversion from PTB to CoNLL. This might be a reason for the high performances of the dependency parsers that directly compute CoNLL dependencies. The results for ENJU-CoNLL and ENJU-PAS show that PAS contributes to a larger accuracy improvement, although this does not necessarily mean the superiority of PAS, because two imperfect conversions, i.e., PAS-to-PTB and PTB-to-CoNLL, are applied for creating CoNLL.

### 4.3 Parser ensemble results

Table 5 shows the accuracy obtained with ensembles of two parsers/representations (except the PTB format). Bracketed figures denote improvements from the accuracy with a single parser/representation. The results show that the task accuracy significantly improves by parser/representation ensemble. Interestingly, the accuracy improvements are observed even for ensembles of different representations from the same parser. This indicates that a single parse representation is insufficient for expressing the true

Bag-of-words features	48.2/54.9/51.1
Yakushiji et al. (2005)	33.7/33.1/33.4
Mitsumori et al. (2006)	54.2/42.6/47.7
Giuliano et al. (2006)	60.9/57.2/59.0
Sætre et al. (2007)	64.3/44.1/52.0
This paper	54.9/65.5/59.5

Table 6: Comparison with previous results on PPI extraction (precision/recall/f-score)

potential of a parser. Effectiveness of the parser ensemble is also attested by the fact that it resulted in larger improvements. Further investigation of the sources of these improvements will illustrate the advantages and disadvantages of these parsers and representations, leading us to better parsing models and a better design for parse representations.

### 4.4 Comparison with previous results on PPI extraction

PPI extraction experiments on AImed have been reported repeatedly, although the figures cannot be compared directly because of the differences in data preprocessing and the number of target protein pairs (Sætre et al., 2007). Table 6 compares our best result with previously reported accuracy figures. Giuliano et al. (2006) and Mitsumori et al. (2006) do not rely on syntactic parsing, while the former applied SVMs with kernels on surface strings and the latter is similar to our baseline method. Bunescu and Mooney (2005) applied SVMs with subsequence kernels to the same task, although they provided only a precision-recall graph, and its f-score is around 50. Since we did not run experiments on protein-pair-wise cross validation, our system cannot be compared directly to the results reported by Erkan et al. (2007) and Katrenko and Adriaans



(2006), while Sætne et al. (2007) presented better results than theirs in the same evaluation criterion.

## 5 Related Work

Though the evaluation of syntactic parsers has been a major concern in the parsing community, and a couple of works have recently presented the comparison of parsers based on different frameworks, their methods were based on the comparison of the parsing accuracy in terms of a certain intermediate parse representation (Ringger et al., 2004; Kaplan et al., 2004; Briscoe and Carroll, 2006; Clark and Curran, 2007; Miyao et al., 2007; Clegg and Shepherd, 2007; Pyysalo et al., 2007b; Pyysalo et al., 2007a; Sagae et al., 2008). Such evaluation requires gold standard data in an intermediate representation. However, it has been argued that the conversion of parsing results into an intermediate representation is difficult and far from perfect.

The relationship between parsing accuracy and task accuracy has been obscure for many years. Quirk and Corston-Oliver (2006) investigated the impact of parsing accuracy on statistical MT. However, this work was only concerned with a single dependency parser, and did not focus on parsers based on different frameworks.

## 6 Conclusion and Future Work

We have presented our attempts to evaluate syntactic parsers and their representations that are based on different frameworks; dependency parsing, phrase structure parsing, or deep parsing. The basic idea is to measure the accuracy improvements of the PPI extraction task by incorporating the parser output as statistical features of a machine learning classifier. Experiments showed that state-of-the-art parsers attain accuracy levels that are on par with each other, while parsing speed differs significantly. We also found that accuracy improvements vary when parsers are retrained with domain-specific data, indicating the importance of domain adaptation and the differences in the portability of parser training methods.

Although we restricted ourselves to parsers trainable with Penn Treebank-style treebanks, our methodology can be applied to any English parsers. Candidates include RASP (Briscoe and Carroll,

2006), the C&C parser (Clark and Curran, 2004), the XLE parser (Kaplan et al., 2004), MINIPAR (Lin, 1998), and Link Parser (Sleator and Temperley, 1993; Pyysalo et al., 2006), but the domain adaptation of these parsers is not straightforward. It is also possible to evaluate unsupervised parsers, which is attractive since evaluation of such parsers with gold-standard data is extremely problematic.

A major drawback of our methodology is that the evaluation is indirect and the results depend on a selected task and its settings. This indicates that different results might be obtained with other tasks. Hence, we cannot conclude the superiority of parsers/representations only with our results. In order to obtain general ideas on parser performance, experiments on other tasks are indispensable.

## Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan), Genome Network Project (MEXT, Japan), and Grant-in-Aid for Young Scientists (MEXT, Japan).

## References

- D. M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.
- T. Briscoe and J. Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *COLING/ACL 2006 Poster Session*.
- R. Bunescu and R. J. Mooney. 2004. Collective information extraction with relational markov networks. In *ACL 2004*, pages 439–446.
- R. C. Bunescu and R. J. Mooney. 2005. Subsequence kernels for relation extraction. In *NIPS 2005*.
- E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL 2005*.
- E. Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL-2000*, pages 132–139.
- S. Clark and J. R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *42nd ACL*.
- S. Clark and J. R. Curran. 2007. Formalism-independent parser evaluation with CCG and DepBank. In *ACL 2007*.
- A. B. Clegg and A. J. Shepherd. 2007. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8:24.

- M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL 2002*.
- M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *35th ACL*.
- M.-C. de Marneffe, B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.
- J. M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996*.
- G. Erkan, A. Ozgur, and D. R. Radev. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *EMNLP 2007*.
- D. Gildea. 2001. Corpus variation and parser performance. In *EMNLP 2001*, pages 167–202.
- C. Giuliano, A. Lavelli, and L. Romano. 2006. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *EACL 2006*.
- T. Hara, Y. Miyao, and J. Tsujii. 2007. Evaluating impact of re-training a lexical disambiguation model on domain adaptation of an HPSG parser. In *IWPT 2007*.
- R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *NODALIDA 2007*.
- R. M. Kaplan, S. Riezler, T. H. King, J. T. Maxwell, and A. Vasserman. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *HLT/NAACL'04*.
- S. Katrenko and P. Adriaans. 2006. Learning relations from biomedical corpora using dependency trees. In *KDECB*, pages 61–80.
- J.-D. Kim, T. Ohta, Y. Teteisi, and J. Tsujii. 2003. GENIA corpus — a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:i180–182.
- D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL 2003*.
- D. Lin. 1998. Dependency-based evaluation of MINIPAR. In *LREC Workshop on the Evaluation of Parsing Systems*.
- M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL 2005*.
- R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL 2006*.
- T. Mitsumori, M. Murata, Y. Fukuda, K. Doi, and H. Doi. 2006. Extracting protein-protein interaction information from biomedical text with SVM. *IEICE - Trans. Inf. Syst.*, E89-D(8):2464–2466.
- Y. Miyao and J. Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.
- Y. Miyao, K. Sagae, and J. Tsujii. 2007. Towards framework-independent evaluation of deep linguistic parsers. In *Grammar Engineering across Frameworks 2007*, pages 238–258.
- A. Moschitti. 2006. Making tree kernels practical for natural language processing. In *EACL 2006*.
- J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *ACL 2005*.
- S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL 2007*.
- S. Pyysalo, T. Salakoski, S. Aubin, and A. Nazarenko. 2006. Lexical adaptation of link grammar to the biomedical sublanguage: a comparative evaluation of three approaches. *BMC Bioinformatics*, 7(Suppl. 3).
- S. Pyysalo, F. Ginter, J. Heimonen, J. Björne, J. Boberg, J. Järvinen, and T. Salakoski. 2007a. BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(50).
- S. Pyysalo, F. Ginter, V. Laippala, K. Haverinen, J. Heimonen, and T. Salakoski. 2007b. On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. In *BioNLP 2007*, pages 25–32.
- C. Quirk and S. Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *EMNLP 2006*.
- E. K. Ringger, R. C. Moore, E. Charniak, L. Vanderwende, and H. Suzuki. 2004. Using the Penn Treebank to evaluate non-treebank parsers. In *LREC 2004*.
- R. Sætre, K. Sagae, and J. Tsujii. 2007. Syntactic features for protein-protein interaction extraction. In *LBM 2007 short papers*.
- K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *EMNLP-CoNLL 2007*.
- K. Sagae, Y. Miyao, T. Matsuzaki, and J. Tsujii. 2008. Challenges in mapping of syntactic representations for framework-independent parser evaluation. In *the Workshop on Automated Syntactic Annotations for Interoperable Language Resources*.
- D. D. Sleator and D. Temperley. 1993. Parsing English with a Link Grammar. In *3rd IWPT*.
- Y. Tsuruoka, Y. Tateishi, J.-D. Kim, T. Ohta, J. McNaught, S. Ananiadou, and J. Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In *10th Panhellenic Conference on Informatics*.
- A. Yakushiji, Y. Miyao, Y. Tateisi, and J. Tsujii. 2005. Biomedical information extraction with predicate-argument structure patterns. In *First International Symposium on Semantic Mining in Biomedicine*.

# MAXSIM: A Maximum Similarity Metric for Machine Translation Evaluation

Yee Seng Chan and Hwee Tou Ng

Department of Computer Science

National University of Singapore

Law Link, Singapore 117590

{chanys, nght}@comp.nus.edu.sg

## Abstract

We propose an automatic machine translation (MT) evaluation metric that calculates a similarity score (based on precision and recall) of a pair of sentences. Unlike most metrics, we compute a similarity score between items across the two sentences. We then find a maximum weight matching between the items such that each item in one sentence is mapped to at most one item in the other sentence. This general framework allows us to use arbitrary similarity functions between items, and to incorporate different information in our comparison, such as n-grams, dependency relations, etc. When evaluated on data from the ACL-07 MT workshop, our proposed metric achieves higher correlation with human judgements than all 11 automatic MT evaluation metrics that were evaluated during the workshop.

## 1 Introduction

In recent years, machine translation (MT) research has made much progress, which includes the introduction of automatic metrics for MT evaluation. Since human evaluation of MT output is time consuming and expensive, having a robust and accurate automatic MT evaluation metric that correlates well with human judgement is invaluable.

Among all the automatic MT evaluation metrics, BLEU (Papineni et al., 2002) is the most widely used. Although BLEU has played a crucial role in the progress of MT research, it is becoming evident that BLEU does not correlate with human judgement

well enough, and suffers from several other deficiencies such as the lack of an intuitive interpretation of its scores.

During the recent ACL-07 workshop on statistical MT (Callison-Burch et al., 2007), a total of 11 automatic MT evaluation metrics were evaluated for correlation with human judgement. The results show that, as compared to BLEU, several recently proposed metrics such as Semantic-role overlap (Gimenez and Marquez, 2007), ParaEval-recall (Zhou et al., 2006), and METEOR (Banerjee and Lavie, 2005) achieve higher correlation.

In this paper, we propose a new automatic MT evaluation metric, MAXSIM, that compares a pair of system-reference sentences by extracting n-grams and dependency relations. Recognizing that different concepts can be expressed in a variety of ways, we allow matching across synonyms and also compute a score between two matching items (such as between two n-grams or between two dependency relations), which indicates their degree of similarity with each other.

Having weighted matches between items means that there could be many possible ways to match, or link items from a system translation sentence to a reference translation sentence. To match each system item to at most one reference item, we model the items in the sentence pair as nodes in a bipartite graph and use the Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957) to find a *maximum* weight matching (or alignment) between the items in polynomial time. The weights (from the edges) of the resulting graph will then be added to determine the final similarity score between the pair of sentences.

Although a maximum weight bipartite graph was also used in the recent work of (Taskar et al., 2005), their focus was on learning supervised models for single word alignment between sentences from a source and target language.

The contributions of this paper are as follows. Current metrics (such as BLEU, METEOR, Semantic-role overlap, ParaEval-recall, etc.) do not assign different weights to their matches: either two items match, or they don't. Also, metrics such as METEOR determine an alignment between the items of a sentence pair by using heuristics such as the least number of matching crosses. In contrast, we propose weighting different matches differently, and then obtain an optimal set of matches, or alignments, by using a maximum weight matching framework. We note that this framework is not used by any of the 11 automatic MT metrics in the ACL-07 MT workshop. Also, this framework allows for defining arbitrary similarity functions between two matching items, and we could match arbitrary concepts (such as dependency relations) gathered from a sentence pair. In contrast, most other metrics (notably BLEU) limit themselves to matching based only on the surface form of words. Finally, when evaluated on the datasets of the recent ACL-07 MT workshop (Callison-Burch et al., 2007), our proposed metric achieves higher correlation with human judgements than all of the 11 automatic MT evaluation metrics evaluated during the workshop.

In the next section, we describe several existing metrics. In Section 3, we discuss issues to consider when designing a metric. In Section 4, we describe our proposed metric. In Section 5, we present our experimental results. Finally, we outline future work in Section 6, before concluding in Section 7.

## 2 Automatic Evaluation Metrics

In this section, we describe BLEU, and the three metrics which achieved higher correlation results than BLEU in the recent ACL-07 MT workshop.

### 2.1 BLEU

BLEU (Papineni et al., 2002) is essentially a precision-based metric and is currently the standard metric for automatic evaluation of MT performance. To score a system translation, BLEU tabulates the

number of n-gram matches of the system translation against one or more reference translations. Generally, more n-gram matches result in a higher BLEU score.

When determining the matches to calculate precision, BLEU uses a *modified*, or *clipped* n-gram precision. With this, an n-gram (from both the system and reference translation) is considered to be exhausted or used after participating in a match. Hence, each system n-gram is "clipped" by the maximum number of times it appears in any reference translation.

To prevent short system translations from receiving too high a score and to compensate for its lack of a recall component, BLEU incorporates a brevity penalty. This penalizes the score of a system if the length of its entire translation output is shorter than the length of the reference text.

### 2.2 Semantic Roles

(Gimenez and Marquez, 2007) proposed using deeper linguistic information to evaluate MT performance. For evaluation in the ACL-07 MT workshop, the authors used the metric which they termed as  $SR-O_r-^*1$ . This metric first counts the number of lexical overlaps  $SR-O_r-t$  for all the different semantic roles  $t$  that are found in the system and reference translation sentence. A uniform average of the counts is then taken as the score for the sentence pair. In their work, the different semantic roles  $t$  they considered include the various core and adjunct arguments as defined in the PropBank project (Palmer et al., 2005). For instance,  $SR-O_r-A0$  refers to the number of lexical overlaps between the  $A0$  arguments. To extract semantic roles from a sentence, several processes such as lemmatization, part-of-speech tagging, base phrase chunking, named entity tagging, and finally semantic role tagging need to be performed.

### 2.3 ParaEval

The ParaEval metric (Zhou et al., 2006) uses a large collection of paraphrases, automatically extracted from parallel corpora, to evaluate MT performance. To compare a pair of sentences, ParaEval first locates paraphrase matches between the two

<sup>1</sup>Verified through personal communication as this is not evident in their paper.

sentences. Then, unigram matching is performed on the remaining words that are not matched using paraphrases. Based on the matches, ParaEval will then elect to use either unigram precision or unigram recall as its score for the sentence pair. In the ACL-07 MT workshop, ParaEval based on recall (ParaEval-recall) achieves good correlation with human judgements.

## 2.4 METEOR

Given a pair of strings to compare (a system translation and a reference translation), METEOR (Banerjee and Lavie, 2005) first creates a word alignment between the two strings. Based on the number of word or unigram matches and the amount of string fragmentation represented by the alignment, METEOR calculates a score for the pair of strings.

In aligning the unigrams, each unigram in one string is mapped, or linked, to at most one unigram in the other string. These word alignments are created incrementally through a series of stages, where each stage only adds alignments between unigrams which have not been matched in previous stages. At each stage, if there are multiple different alignments, then the alignment with the most number of mappings is selected. If there is a tie, then the alignment with the least number of unigram mapping crosses is selected.

The three stages of “exact”, “porter stem”, and “WN synonymy” are usually applied in sequence to create alignments. The “exact” stage maps unigrams if they have the same surface form. The “porter stem” stage then considers the remaining unmapped unigrams and maps them if they are the same after applying the Porter stemmer. Finally, the “WN synonymy” stage considers all remaining unigrams and maps two unigrams if they are synonyms in the WordNet sense inventory (Miller, 1990).

Once the final alignment has been produced, unigram precision  $P$  (number of unigram matches  $m$  divided by the total number of system unigrams) and unigram recall  $R$  ( $m$  divided by the total number of reference unigrams) are calculated and combined into a single parameterized harmonic mean (Rijsbergen, 1979):

$$F_{mean} = \frac{P \cdot R}{\alpha P + (1 - \alpha)R} \quad (1)$$

To account for longer matches and the amount of fragmentation represented by the alignment, METEOR groups the matched unigrams into as few *chunks* as possible and imposes a penalty based on the number of chunks. The METEOR score for a pair of sentences is:

$$score = \left[ 1 - \gamma \left( \frac{\text{no. of chunks}}{m} \right)^\beta \right] F_{mean}$$

where  $\gamma \left( \frac{\text{no. of chunks}}{m} \right)^\beta$  represents the fragmentation penalty of the alignment. Note that METEOR consists of three parameters that need to be optimized based on experimentation:  $\alpha$ ,  $\beta$ , and  $\gamma$ .

## 3 Metric Design Considerations

We first review some aspects of existing metrics and highlight issues that should be considered when designing an MT evaluation metric.

- **Intuitive interpretation:** To compensate for the lack of recall, BLEU incorporates a brevity penalty. This, however, prevents an intuitive interpretation of its scores. To address this, standard measures like precision and recall could be used, as in some previous research (Banerjee and Lavie, 2005; Melamed et al., 2003).
- **Allowing for variation:** BLEU only counts exact word matches. Languages, however, often allow a great deal of variety in vocabulary and in the ways concepts are expressed. Hence, using information such as synonyms or dependency relations could potentially address the issue better.
- **Matches should be weighted:** Current metrics either match, or don’t match a pair of items. We note, however, that matches between items (such as words, n-grams, etc.) should be weighted according to their *degree* of similarity.

## 4 The Maximum Similarity Metric

We now describe our proposed metric, Maximum Similarity (MAXSIM), which is based on precision and recall, allows for synonyms, and weights the matches found.

Given a pair of English sentences to be compared (a system translation against a reference translation), we perform tokenization<sup>2</sup>, lemmatization using WordNet<sup>3</sup>, and part-of-speech (POS) tagging with the MXPOST tagger (Ratnaparkhi, 1996). Next, we remove all non-alphanumeric tokens. Then, we match the unigrams in the system translation to the unigrams in the reference translation. Based on the matches, we calculate the recall and precision, which we then combine into a single  $F_{mean}$  unigram score using Equation 1. Similarly, we also match the bigrams and trigrams of the sentence pair and calculate their corresponding  $F_{mean}$  scores. To obtain a single similarity score  $score_s$  for this sentence pair  $s$ , we simply average the three  $F_{mean}$  scores. Then, to obtain a single similarity score  $sim-score$  for the entire system corpus, we repeat this process of calculating a  $score_s$  for each system-reference sentence pair  $s$ , and compute the average over all  $|S|$  sentence pairs:

$$sim-score = \frac{1}{|S|} \sum_{s=1}^{|S|} \left[ \frac{1}{N} \sum_{n=1}^N F_{mean_{s,n}} \right]$$

where in our experiments, we set  $N=3$ , representing calculation of unigram, bigram, and trigram scores. If we are given access to multiple references, we calculate an individual  $sim-score$  between the system corpus and *each* reference corpus, and then average the scores obtained.

#### 4.1 Using N-gram Information

In this subsection, we describe in detail how we match the n-grams of a system-reference sentence pair.

**Lemma and POS match** Representing each n-gram by its sequence of lemma and POS-tag pairs, we first try to perform an exact match in both lemma and POS-tag. In all our n-gram matching, each n-gram in the system translation can only match at most *one* n-gram in the reference translation.

Representing each unigram ( $l_i p_i$ ) at position  $i$  by its lemma  $l_i$  and POS-tag  $p_i$ , we count the number  $match_{uni}$  of system-reference unigram pairs where both their lemma and POS-tag match. To find matching pairs, we proceed in a left-to-right fashion

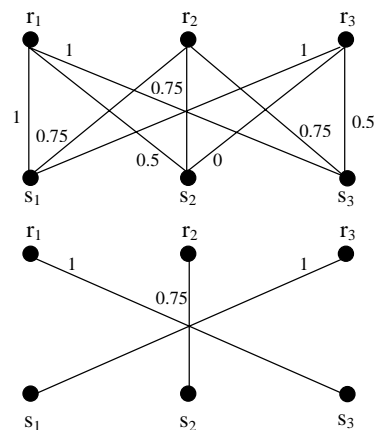


Figure 1: Bipartite matching.

(in both strings). We first compare the first system unigram to the first reference unigram, then to the second reference unigram, and so on until we find a match. If there is a match, we increment  $match_{uni}$  by 1 and remove this pair of system-reference unigrams from further consideration (removed items will not be matched again subsequently). Then, we move on to the second system unigram and try to match it against the reference unigrams, once again proceeding in a left-to-right fashion. We continue this process until we reach the last system unigram.

To determine the number  $match_{bi}$  of bigram matches, a system bigram ( $l_{s_i} p_{s_i}, l_{s_{i+1}} p_{s_{i+1}}$ ) matches a reference bigram ( $l_{r_i} p_{r_i}, l_{r_{i+1}} p_{r_{i+1}}$ ) if  $l_{s_i} = l_{r_i}$ ,  $p_{s_i} = p_{r_i}$ ,  $l_{s_{i+1}} = l_{r_{i+1}}$ , and  $p_{s_{i+1}} = p_{r_{i+1}}$ . For trigrams, we similarly determine  $match_{tri}$  by counting the number of trigram matches.

**Lemma match** For the remaining set of n-grams that are not yet matched, we now relax our matching criteria by allowing a match if their corresponding lemmas match. That is, a system unigram ( $l_{s_i} p_{s_i}$ ) matches a reference unigram ( $l_{r_i} p_{r_i}$ ) if  $l_{s_i} = l_{r_i}$ . In the case of bigrams, the matching conditions are  $l_{s_i} = l_{r_i}$  and  $l_{s_{i+1}} = l_{r_{i+1}}$ . The conditions for trigrams are similar. Once again, we find matches in a left-to-right fashion. We add the number of unigram, bigram, and trigram matches found during this phase to  $match_{uni}$ ,  $match_{bi}$ , and  $match_{tri}$  respectively.

**Bipartite graph matching** For the remaining n-grams that are not matched so far, we try to match them by constructing bipartite graphs. During this phase, we will construct three bipartite graphs, one

<sup>2</sup><http://www.cis.upenn.edu/~treebank/tokenizer.sed>

<sup>3</sup><http://wordnet.princeton.edu/man/morph.3WN>

each for the remaining set of unigrams, bigrams, and trigrams.

Using bigrams to illustrate, we construct a weighted complete bipartite graph, where each edge  $e$  connecting a pair of system-reference bigrams has a weight  $w(e)$ , indicating the degree of similarity between the bigrams connected. Note that, without loss of generality, if the number of system nodes and reference nodes (bigrams) are not the same, we can simply add dummy nodes with connecting edges of weight 0 to obtain a complete bipartite graph with equal number of nodes on both sides.

In an  $n$ -gram bipartite graph, the similarity score, or the weight  $w(e)$  of the edge  $e$  connecting a system  $n$ -gram  $(l_{s_1}p_{s_1}, \dots, l_{s_n}p_{s_n})$  and a reference  $n$ -gram  $(l_{r_1}p_{r_1}, \dots, l_{r_n}p_{r_n})$  is calculated as follows:

$$S_i = \frac{I(p_{s_i}, p_{r_i}) + \text{Syn}(l_{s_i}, l_{r_i})}{2}$$

$$w(e) = \frac{1}{n} \sum_{i=1}^n S_i$$

where  $I(p_{s_i}, p_{r_i})$  evaluates to 1 if  $p_{s_i} = p_{r_i}$ , and 0 otherwise. The function  $\text{Syn}(l_{s_i}, l_{r_i})$  checks whether  $l_{s_i}$  is a synonym of  $l_{r_i}$ . To determine this, we first obtain the set  $WN_{\text{syn}}(l_{s_i})$  of WordNet synonyms for  $l_{s_i}$  and the set  $WN_{\text{syn}}(l_{r_i})$  of WordNet synonyms for  $l_{r_i}$ . Then,

$$\text{Syn}(l_{s_i}, l_{r_i}) = \begin{cases} 1, & WN_{\text{syn}}(l_{s_i}) \cap WN_{\text{syn}}(l_{r_i}) \\ & \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

In gathering the set  $WN_{\text{syn}}$  for a word, we gather all the synonyms for all its senses and do not restrict to a particular POS category. Further, if we are comparing bigrams or trigrams, we impose an additional condition:  $S_i \neq 0$ , for  $1 \leq i \leq n$ , else we will set  $w(e) = 0$ . This captures the intuition that in matching a system  $n$ -gram against a reference  $n$ -gram, where  $n > 1$ , we require each system token to have at least some degree of similarity with the corresponding reference token.

In the top half of Figure 1, we show an example of a complete bipartite graph, constructed for a set of three system bigrams  $(s_1, s_2, s_3)$  and three reference bigrams  $(r_1, r_2, r_3)$ , and the weight of the connecting edge between two bigrams represents their degree of similarity.

Next, we aim to find a *maximum* weight matching (or alignment) between the bigrams such that each system (reference) bigram is connected to exactly one reference (system) bigram. This *maximum weighted bipartite matching* problem can be solved in  $O(n^3)$  time (where  $n$  refers to the number of nodes, or vertices in the graph) using the Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957). The bottom half of Figure 1 shows the resulting maximum weighted bipartite graph, where the alignment represents the maximum weight matching, out of all possible alignments.

Once we have solved and obtained a maximum weight matching  $M$  for the bigram bipartite graph, we sum up the weights of the edges to obtain the weight of the matching  $M$ :  $w(M) = \sum_{e \in M} w(e)$ , and add  $w(M)$  to  $match_{bi}$ . From the unigram and trigram bipartite graphs, we similarly calculate their respective  $w(M)$  and add to the corresponding  $match_{uni}$  and  $match_{tri}$ .

Based on  $match_{uni}$ ,  $match_{bi}$ , and  $match_{tri}$ , we calculate their corresponding precision  $P$  and recall  $R$ , from which we obtain their respective  $F_{mean}$  scores via Equation 1. Using bigrams for illustration, we calculate its  $P$  and  $R$  as:

$$P = \frac{match_{bi}}{\text{no. of bigrams in system translation}}$$

$$R = \frac{match_{bi}}{\text{no. of bigrams in reference translation}}$$

## 4.2 Dependency Relations

Besides matching a pair of system-reference sentences based on the surface form of words, previous work such as (Gimenez and Marquez, 2007) and (Rajman and Hartley, 2002) had shown that deeper linguistic knowledge such as semantic roles and syntax can be usefully exploited.

In the previous subsection, we describe our method of using bipartite graphs for matching of  $n$ -grams found in a sentence pair. This use of bipartite graphs, however, is a very general framework to obtain an optimal alignment of the corresponding ‘‘information items’’ contained within a sentence pair. Hence, besides matching based on  $n$ -gram strings, we can also match other ‘‘information items’’, such as dependency relations.

Metric	Adequacy	Fluency	Rank	Constituent	Average
MAXSIM <sub>n+d</sub>	0.780	0.827	0.875	0.760	0.811
MAXSIM <sub>n</sub>	<b>0.804</b>	<b>0.845</b>	<b>0.893</b>	0.766	<b>0.827</b>
Semantic-role	0.774	0.839	0.804	0.742	0.790
ParaEval-recall	0.712	0.742	0.769	<b>0.798</b>	0.755
METEOR	0.701	0.719	0.746	0.670	0.709
BLEU	0.690	0.722	0.672	0.603	0.672

Table 1: Overall correlations on the Europarl and News Commentary datasets. The ‘Semantic-role overlap’ metric is abbreviated as ‘Semantic-role’. Note that each figure above represents 6 translation tasks: the Europarl and News Commentary datasets each with 3 language pairs (German-English, Spanish-English, French-English).

In our work, we train the MSTParser<sup>4</sup> (McDonald et al., 2005) on the Penn Treebank Wall Street Journal (WSJ) corpus, and use it to extract dependency relations from a sentence. Currently, we focus on extracting only two relations: *subject* and *object*. For each relation ( $ch, dp, pa$ ) extracted, we note the child lemma  $ch$  of the relation (often a noun), the relation type  $dp$  (either *subject* or *object*), and the parent lemma  $pa$  of the relation (often a verb). Then, using the system relations and reference relations extracted from a system-reference sentence pair, we similarly construct a bipartite graph, where each node is a relation ( $ch, dp, pa$ ). We define the weight  $w(e)$  of an edge  $e$  between a system relation ( $ch_s, dp_s, pa_s$ ) and a reference relation ( $ch_r, dp_r, pa_r$ ) as follows:

$$\frac{Syn(ch_s, ch_r) + I(dp_s, dp_r) + Syn(pa_s, pa_r)}{3}$$

where functions  $I$  and  $Syn$  are defined as in the previous subsection. Also,  $w(e)$  is non-zero only if  $dp_s = dp_r$ . After solving for the maximum weight matching  $M$ , we divide  $w(M)$  by the number of system relations extracted to obtain a precision score  $P$ , and divide  $w(M)$  by the number of reference relations extracted to obtain a recall score  $R$ .  $P$  and  $R$  are then similarly combined into a  $F_{mean}$  score for the sentence pair. To compute the similarity score when incorporating dependency relations, we average the  $F_{mean}$  scores for unigrams, bigrams, trigrams, and dependency relations.

## 5 Results

To evaluate our metric, we conduct experiments on datasets from the ACL-07 MT workshop and NIST

<sup>4</sup>Available at: <http://sourceforge.net/projects/mstparser>

Europarl					
Metric	Adq	Flu	Rank	Con	Avg
MAXSIM <sub>n+d</sub>	0.749	0.786	<b>0.857</b>	0.651	<b>0.761</b>
MAXSIM <sub>n</sub>	0.749	0.786	<b>0.857</b>	0.651	<b>0.761</b>
Semantic-role	<b>0.815</b>	<b>0.854</b>	0.759	0.612	0.760
ParaEval-recall	0.701	0.708	0.737	<b>0.772</b>	0.730
METEOR	0.726	0.741	0.770	0.558	0.699
BLEU	0.803	0.822	0.699	0.512	0.709

Table 2: Correlations on the Europarl dataset. Adq=Adequacy, Flu=Fluency, Con=Constituent, and Avg=Average.

News Commentary					
Metric	Adq	Flu	Rank	Con	Avg
MAXSIM <sub>n+d</sub>	0.812	0.869	0.893	0.869	0.861
MAXSIM <sub>n</sub>	<b>0.860</b>	<b>0.905</b>	<b>0.929</b>	<b>0.881</b>	<b>0.894</b>
Semantic-role	0.734	0.824	0.848	0.871	0.819
ParaEval-recall	0.722	0.777	0.800	0.824	0.781
METEOR	0.677	0.698	0.721	0.782	0.720
BLEU	0.577	0.622	0.646	0.693	0.635

Table 3: Correlations on the News Commentary dataset.

MT 2003 evaluation exercise.

### 5.1 ACL-07 MT Workshop

The ACL-07 MT workshop evaluated the translation quality of MT systems on various translation tasks, and also measured the correlation (with human judgement) of 11 automatic MT evaluation metrics. The workshop used a Europarl dataset and a News Commentary dataset, where each dataset consisted of English sentences (2,000 English sentences for Europarl and 2,007 English sentences for News Commentary) and their translations in various languages. As part of the workshop, correlations of the automatic metrics were measured for the tasks



of translating German, Spanish, and French into English. Hence, we will similarly measure the correlation of MAXSIM on these tasks.

### 5.1.1 Evaluation Criteria

For human evaluation of the MT submissions, four different criteria were used in the workshop: **Adequacy** (how much of the original meaning is expressed in a system translation), **Fluency** (the translation’s fluency), **Rank** (different translations of a single source sentence are compared and ranked from best to worst), and **Constituent** (some constituents from the parse tree of the source sentence are translated, and human judges have to rank these translations).

During the workshop, Kappa values measured for inter- and intra-annotator agreement for *rank* and *constituent* are substantially higher than those for *adequacy* and *fluency*, indicating that *rank* and *constituent* are more reliable criteria for MT evaluation.

### 5.1.2 Correlation Results

We follow the ACL-07 MT workshop process of converting the raw scores assigned by an automatic metric to ranks and then using the Spearman’s rank correlation coefficient to measure correlation.

During the workshop, only three automatic metrics (Semantic-role overlap, ParaEval-recall, and METEOR) achieve higher correlation than BLEU. We gather the correlation results of these metrics from the workshop paper (Callison-Burch et al., 2007), and show in Table 1 the overall correlations of these metrics over the Europarl and News Commentary datasets. In the table, MAXSIM<sub>n</sub> represents using only n-gram information (Section 4.1) for our metric, while MAXSIM<sub>n+d</sub> represents using both n-gram and dependency information. We also show the breakdown of the correlation results into the Europarl dataset (Table 2) and the News Commentary dataset (Table 3). In all our results for MAXSIM in this paper, we follow METEOR and use  $\alpha=0.9$  (weighing recall more than precision) in our calculation of  $F_{mean}$  via Equation 1, unless otherwise stated.

The results in Table 1 show that MAXSIM<sub>n</sub> and MAXSIM<sub>n+d</sub> achieve overall average (over the four criteria) correlations of 0.827 and 0.811 respectively. Note that these results are substantially

Metric	Adq	Flu	Avg
MAXSIM <sub>n+d</sub>	0.943	0.886	0.915
MAXSIM <sub>n</sub>	0.829	0.771	0.800
METEOR (optimized)	1.000	0.943	0.972
METEOR	0.943	0.886	0.915
BLEU	0.657	0.543	0.600

Table 4: Correlations on the NIST MT 2003 dataset.

higher than BLEU, and in particular higher than the *best* performing *Semantic-role overlap* metric in the ACL-07 MT workshop. Also, Semantic-role overlap requires more processing steps (such as base phrase chunking, named entity tagging, etc.) than MAXSIM. For future work, we could experiment with incorporating semantic-role information into our current framework. We note that the ParaEval-recall metric achieves higher correlation on the *constituent* criterion, which might be related to the fact that both ParaEval-recall and the *constituent* criterion are based on phrases: ParaEval-recall tries to match phrases, and the *constituent* criterion is based on judging translations of phrases.

## 5.2 NIST MT 2003 Dataset

We also conduct experiments on the test data (LDC2006T04) of NIST MT 2003 Chinese-English translation task. For this dataset, human judgements are available on *adequacy* and *fluency* for six system submissions, and there are four English reference translation texts.

Since implementations of the BLEU and METEOR metrics are publicly available, we score the system submissions using BLEU (version 11b with its default settings), METEOR, and MAXSIM, showing the resulting correlations in Table 4. For METEOR, when used with its originally proposed parameter values of ( $\alpha=0.9$ ,  $\beta=3.0$ ,  $\gamma=0.5$ ), which the METEOR researchers mentioned were based on some early experimental work (Banerjee and Lavie, 2005), we obtain an average correlation value of 0.915, as shown in the row “METEOR”. In the recent work of (Lavie and Agarwal, 2007), the values of these parameters were tuned to be ( $\alpha=0.81$ ,  $\beta=0.83$ ,  $\gamma=0.28$ ), based on experiments on the NIST 2003 and 2004 Arabic-English evaluation datasets. When METEOR was run with these new parameter values, it returned an average correlation value of

0.972, as shown in the row “METEOR (optimized)”.

MAXSIM using only n-gram information (MAXSIM<sub>n</sub>) gives an average correlation value of 0.800, while adding dependency information (MAXSIM<sub>n+d</sub>) improves the correlation value to 0.915. Note that so far, the parameters of MAXSIM are not optimized and we simply perform uniform averaging of the different n-grams and dependency scores. Under this setting, the correlation achieved by MAXSIM is comparable to that achieved by METEOR.

## 6 Future Work

In our current work, the parameters of MAXSIM are as yet un-optimized. We found that by setting  $\alpha=0.7$ , MAXSIM<sub>n+d</sub> could achieve a correlation of 0.972 on the NIST MT 2003 dataset. Also, we have barely exploited the potential of weighted similarity matching. Possible future directions include adding semantic role information, using the distance between item pairs based on the token position within each sentence as additional weighting consideration, etc. Also, we have seen that dependency relations help to improve correlation on the NIST dataset, but not on the ACL-07 MT workshop datasets. Since the accuracy of dependency parsers is not perfect, a possible future work is to identify when best to incorporate such syntactic information.

## 7 Conclusion

In this paper, we present MAXSIM, a new automatic MT evaluation metric that computes a similarity score between corresponding items across a sentence pair, and uses a bipartite graph to obtain an optimal matching between item pairs. This general framework allows us to use arbitrary similarity functions between items, and to incorporate different information in our comparison. When evaluated for correlation with human judgements, MAXSIM achieves superior results when compared to current automatic MT evaluation metrics.

## References

S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the*

*Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization, ACL05*, pages 65–72.

- C. Callison-Burch, C. Fordyce, P. Koehn, C. Monz, and J. Schroeder. 2007. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation, ACL07*, pages 136–158.
- J. Gimenez and L. Marquez. 2007. Linguistic features for automatic evaluation of heterogeneous MT systems. In *Proceedings of the Second Workshop on Statistical Machine Translation, ACL07*, pages 256–264.
- H. W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2(1):83–97.
- A. Lavie and A. Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation, ACL07*, pages 228–231.
- R. McDonald, K. Crammer, and F. Pereira. 2005. On-line large-margin training of dependency parsers. In *Proceedings of ACL05*, pages 91–98.
- I. D. Melamed, R. Green, and J. P. Turian. 2003. Precision and recall of machine translation. In *Proceedings of HLT-NAACL03*, pages 61–63.
- G. A. Miller. 1990. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.
- J. Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.
- M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.
- K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL02*, pages 311–318.
- M. Rajman and A. Hartley. 2002. Automatic ranking of MT systems. In *Proceedings of LREC02*, pages 1247–1253.
- A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP96*, pages 133–142.
- C. Rijsbergen. 1979. *Information Retrieval*. Butterworths, London, UK, 2nd edition.
- B. Taskar, S. Lacoste-Julien, and D. Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of HLT/EMNLP05*, pages 73–80.
- L. Zhou, C. Y. Lin, and E. Hovy. 2006. Re-evaluating machine translation results with paraphrase support. In *Proceedings of EMNLP06*, pages 77–84.

# Contradictions and Justifications: Extensions to the Textual Entailment Task

Ellen M. Voorhees

National Institute of Standards and Technology  
Gaithersburg, MD 20899-8940, USA  
ellen.voorhees@nist.gov

## Abstract

The third PASCAL Recognizing Textual Entailment Challenge (RTE-3) contained an optional task that extended the main entailment task by requiring a system to make three-way entailment decisions (entails, contradicts, neither) and to justify its response. Contradiction was rare in the RTE-3 test set, occurring in only about 10% of the cases, and systems found accurately detecting it difficult. Subsequent analysis of the results shows a test set must contain many more entailment pairs for the three-way decision task than the traditional two-way task to have equal confidence in system comparisons. Each of six human judges representing eventual end users rated the quality of a justification by assigning “understandability” and “correctness” scores. Ratings of the same justification across judges differed significantly, signaling the need for a better characterization of the justification task.

## 1 Introduction

The PASCAL Recognizing Textual Entailment (RTE) workshop series (see [www.pascal-network.org/Challenges/RTE3/](http://www.pascal-network.org/Challenges/RTE3/)) has been a catalyst for recent research in developing systems that are able to detect when the content of one piece of text necessarily follows from the content of another piece of text (Dagan et al., 2006; Giampiccolo et al., 2007). This ability is seen as a fundamental component in the solutions for a variety of natural language problems such as question answering, summarization, and information extraction. In addition

to the main entailment task, the most recent Challenge, RTE-3, contained a second optional task that extended the main task in two ways. The first extension was to require systems to make three-way entailment decisions; the second extension was for systems to return a justification or explanation of how its decision was reached.

In the main RTE entailment task, systems report whether the *hypothesis* is entailed by the *text*. The system responds with YES if the hypothesis is entailed and NO otherwise. But this binary decision conflates the case when the hypothesis actually contradicts the text—the two could not both be true—with simple lack of entailment. The three-way entailment decision task requires systems to decide whether the hypothesis is entailed by the text (YES), contradicts the text (NO), or is neither entailed by nor contradicts the text (UNKNOWN).

The second extension required a system to explain why it reached its conclusion in terms suitable for an eventual end user (i.e., not system developer). Explanations are one way to build a user’s trust in a system, but it is not known what kinds of information must be conveyed nor how best to present that information. RTE-3 provided an opportunity to collect a diverse sample of explanations to begin to explore these questions.

This paper analyzes the extended task results, with the next section describing the three-way decision subtask and Section 3 the justification subtask. Contradiction was rare in the RTE-3 test set, occurring in only about 10% of the cases, and systems found accurately detecting it difficult. While the level of agreement among human annotators as to

the correct answer for an entailment pair was within expected bounds, the test set was found to be too small to reliably distinguish among systems’ three-way accuracy scores. Human judgments of the quality of a justification varied widely, signaling the need for a better characterization of the justification task. Comments from the judges did include some common themes. Judges prized conciseness, though they were uncomfortable with mathematical notation unless they had a mathematical background. Judges strongly disliked being shown system internals such as scores reported by various components.

## 2 The Three-way Decision Task

The extended task used the RTE-3 main task test set of entailment pairs as its test set. This test set contains 800 text and hypothesis pairs, roughly evenly split between pairs for which the text entails the hypothesis (410 pairs) and pairs for which it does not (390 pairs), as defined by the reference answer key released by RTE organizers.

RTE uses an “ordinary understanding” principle for deciding entailment. The hypothesis is considered entailed by the text if a human reading the text would most likely conclude that the hypothesis were true, even if there could exist unusual circumstances that would invalidate the hypothesis. It is explicitly acknowledged that ordinary understanding depends on a common human understanding of language as well as common background knowledge. The extended task also used the ordinary understanding principle for deciding contradictions. The hypothesis and text were deemed to contradict if a human would most likely conclude that the text and hypothesis could not both be true.

The answer key for the three-way decision task was developed at the National Institute of Standards and Technology (NIST) using annotators who had experience as TREC and DUC assessors. NIST assessors annotated all 800 entailment pairs in the test set, with each pair independently annotated by two different assessors. The three-way answer key was formed by keeping exactly the same set of YES answers as in the two-way key (regardless of the NIST annotations) and having NIST staff adjudicate assessor differences on the remainder. This resulted in a three-way answer key containing 410 (51%)

Reference Answer	Systems’ Responses			Totals
	YES	UNKN	NO	
YES	2449	2172	299	4920
UNKN	929	2345	542	3816
NO	348	415	101	864
Totals	3726	4932	942	9600

Table 1: Contingency table of responses over all 800 entailment pairs and all 12 runs.

YES answers, 319 (40%) UNKNOWN answers, and 72 (9%) NO answers.

### 2.1 System results

Eight different organizations participated in the three-way decision subtask submitting a total of 12 runs. A run consists of exactly one response of YES, NO, or UNKNOWN for each of the 800 test pairs. Runs were evaluated using accuracy, the percentage of system responses that match the reference answer.

Figure 1 shows both the overall accuracy of each of the runs (numbers running along the top of the graph) and the accuracy as conditioned on the reference answer (bars). The conditioned accuracy for YES answers, for example, is accuracy computed using just those test pairs for which YES is the reference answer. The runs are sorted by decreasing overall accuracy.

Systems were much more accurate in recognizing entailment than contradiction (black bars are greater than white bars). Since conditioned accuracy does not penalize for overgeneration of a response, the conditioned accuracy for UNKNOWN is excellent for those systems that used UNKNOWN as their default response. Run H never concluded that a pair was a contradiction, for example.

Table 1 gives another view of the relative difficulty of detecting contradiction. The table is a contingency table of the systems’ responses versus the reference answer summed over all test pairs and all runs. A reference answer is represented as a row in the table and a system’s response as a column. Since there are 800 pairs in the test set and 12 runs, there is a total of 9600 responses.

As a group the systems returned NO as a response 942 times, approximately 10% of the time. While 10% is a close match to the 9% of the test set for which NO is the reference answer, the systems detected contradictions for the wrong pairs: the table’s

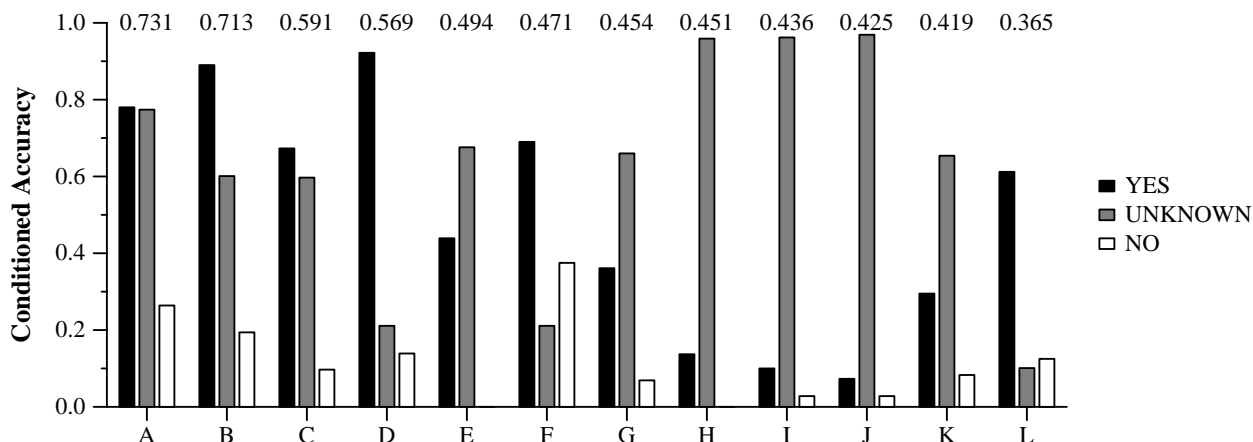


Figure 1: Overall accuracy (top number) and accuracy conditioned by reference answer for three-way runs.

diagonal entry for NO is the smallest entry in both its row and its column. The smallest row entry means that systems were more likely to respond that the hypothesis was entailed than that it contradicted when it in fact contradicted. The smallest column entry means that when the systems did respond that the hypothesis contradicted, it was more often the case that the hypothesis was actually entailed than that it contradicted. The 101 correct NO responses represent 12% of the 864 possible correct NO responses. In contrast, the systems responded correctly for 50% (2449/4920) of the cases when YES was the reference answer and for 61% (2345/3816) of the cases when UNKNOWN was the reference answer.

## 2.2 Human agreement

Textual entailment is evaluated assuming that there is a single correct answer for each test pair. This is a simplifying assumption used to make the evaluation tractable, but as with most NLP phenomena it is not actually true. It is quite possible for two humans to have legitimate differences of opinions (i.e., to differ when neither is mistaken) about whether a hypothesis is entailed or contradicts, especially given annotations are based on ordinary understanding.

Since systems are given credit only when they respond with the reference answer, differences in annotators' opinions can clearly affect systems' accuracy scores. The RTE main task addressed this issue by including a candidate entailment pair in the test set only if multiple annotators agreed on its disposition (Giampiccolo et al., 2007). The test set also

Main Task	NIST Judge 1		
	YES	UNKN	NO
YES	378	27	5
NO	48	242	100
<i>conflated agreement = .90</i>			
Main Task	NIST Judge 2		
	YES	UNKN	NO
YES	383	23	4
NO	46	267	77
<i>conflated agreement = .91</i>			

Table 2: Agreement between NIST judges (columns) and main task reference answers (rows).

contains 800 pairs so an individual test case contributes only  $1/800 = 0.00125$  to the overall accuracy score. To allow the results from the two- and three-way decision tasks to be comparable (and to leverage the cost of creating the main task test set), the extended task used the same test set as the main task and used simple accuracy as the evaluation measure. The expectation was that this would be as effective an evaluation design for the three-way task as it is for the two-way task. Unfortunately, subsequent analysis demonstrates that this is not so.

Recall that NIST judges annotated all 800 entailment pairs in the test set, with each pair independently annotated twice. For each entailment pair, one of the NIST judges was arbitrarily assigned as the first judge for that pair and the other as the second judge. The agreement between NIST and RTE annotators is shown in Table 2. The top half of

the table shows the agreement between the two-way answer key and the annotations of the set of first judges; the bottom half is the same except using the annotations of the set of second judges. The NIST judges’ answers are given in the columns and the two-way reference answers in the rows. Each cell in the table gives the raw count before adjudication of the number of test cases that were assigned that combination of annotations. Agreement is then computed as the percentage of matches when a NIST judge’s NO or UNKNOWN annotation matched a NO two-way reference answer. Agreement is essentially identical for both sets of judges at 0.90 and 0.91 respectively.

Because the agreement numbers reflect the raw counts before adjudication, at least some of the differences may be attributable to annotator errors that were corrected during adjudication. But there do exist legitimate differences of opinion, even for the extreme cases of entails versus contradicts. Typical disagreements involve granularity of place names and amount of background knowledge assumed. Example disagreements concerned whether Hollywood was equivalent to Los Angeles, whether East Jerusalem was equivalent to Jerusalem, and whether members of the same political party who were at odds with one another were ‘opponents’.

RTE organizers reported an agreement rate of about 88% among their annotators for the two-way task (Giampiccolo et al., 2007). The 90% agreement rate between the NIST judges and the two-way answer key probably reflects a somewhat larger amount of disagreement since the test set already had RTE annotators’ disagreements removed. But it is similar enough to support the claim that the NIST annotators agree with other annotators as often as can be expected. Table 3 shows the three-way agreement between the two NIST annotators. As above, the table gives the raw counts before adjudication and agreement is computed as percentage of matching annotations. Three-way agreement is 0.83—smaller than two-way agreement simply because there are more ways to disagree.

Just as annotator agreement declines as the set of possible answers grows, the inherent stability of the accuracy measure also declines: accuracy and agreement are both defined as the percentage of exact matches on answers. The increased uncertainty

	YES	UNKN	NO
YES	381		
UNKN	82	217	
NO	11	43	66

*three-way agreement = .83*

Table 3: Agreement between NIST judges.

when moving from two-way to three-way decisions significantly reduces the power of the evaluation. With the given level of annotator agreement and 800 pairs in the test set, in theory accuracy scores could change by as much as  $136 \text{ (the number of test cases for which annotators disagreed)} \times 0.00125 = .17$  by using a different choice of annotator. The maximum difference in accuracy scores actually observed in the submitted runs was 0.063.

Previous analyses of other evaluation tasks such as document retrieval and question answering demonstrated that system rankings are stable despite differences of opinion in the underlying annotations (Voorhees, 2000; Voorhees and Tice, 2000). The differences in accuracy observed for the three-way task are large enough to affect system rankings, however. Compared to the system ranking of ABCDEFGHIJKL induced by the official three-way answer key, the ranking induced by the first set of judges’ raw annotations is BADCFEGKHLIJ. The ranking induced by the second set of judges’ raw annotations is much more similar to the official results, ABCDEFGHKIJL.

How then to proceed? Since the three-way decision task was motivated by the belief that distinguishing contradiction from simple non-entailment is important, reverting back to a binary decision task is not an attractive option. Increasing the size of the test set beyond 800 test cases will result in a more stable evaluation, though it is not known how big the test set needs to be. Defining new annotation rules in hopes of increasing annotator agreement is a satisfactory option only if those rules capture a characteristic of entailment that systems should actually embody. Reasonable people *do* disagree about entailment and it is unwise to enforce some arbitrary definition in the name of consistency. Using UNKNOWN as the reference answer for all entailment pairs on which annotators disagree may be a reasonable strategy: the disagreement itself is strong evidence that

neither of the other options holds. Creating balanced test sets using this rule could be difficult, however. Following this rule, the RTE-3 test set would have 360 (45%) YES answers, 64 (8%) NO answers, and 376 (47%) UNKNOWN answers, and would induce the ranking ABCDEHIJGKFL. (Runs such as H, I, and J that return UNKNOWN as a default response are rewarded using this annotation rule.)

### 3 Justifications

The second part of the extended task was for systems to provide explanations of how they reached their conclusions. The specification of a justification for the purposes of the task was deliberately vague—a collection of ASCII strings with no minimum or maximum size—so as to not preclude good ideas by arbitrary rules. A justification run contained all of the information from a three-way decision run plus the rationale explaining the response for each of the 800 test pairs in the RTE-3 test set. Six of the runs shown in Figure 1 (A, B, C, D, F, and H) are justification runs. Run A is a manual justification run, meaning there was some human tweaking of the justifications (but not the entailment decisions).

After the runs were submitted, NIST selected a subset of 100 test pairs to be used in the justification evaluation. The pairs were selected by NIST staff after looking at the justifications so as to maximize the informativeness of the evaluation set. All runs were evaluated on the same set of 100 pairs.

Figure 2 shows the justification produced by each run for pair 75 (runs D and F were submitted by the same organization and contained identical justifications for many pairs including pair 75). The text of pair 75 is *Muybridge had earlier developed an invention he called the Zoopraxiscope.*, and the hypothesis is *The Zoopraxiscope was invented by Muybridge.* The hypothesis is entailed by the text, and each of the systems correctly replied that it is entailed. Explanations for why the hypothesis is entailed differ widely, however, with some rationales of dubious validity.

Each of the six different NIST judges rated all 100 justifications. For a given justification, a judge first assigned an integer score between 1–5 on how understandable the justification was (with 1 as unintelligible and 5 as completely understandable). If the

understandability score assigned was 3 or greater, the judge then assigned a correctness score, also an integer between 1–5 with 5 the high score. This second score was interpreted as how compelling the argument contained in the justification was rather than simple correctness because justifications could be strictly correct but immaterial.

#### 3.1 System results

The motivation for the justification subtask was to gather data on how systems might best explain themselves to eventual end users. Given this goal and the exploratory nature of the exercise, judges were given minimal guidance on how to assign scores other than that it should be from a user’s, not a system developer’s, point of view. Judges used a system that displayed the text, hypothesis, and reference answer, and then displayed each submission’s justification in turn. The order in which the runs’ justifications were displayed was randomly selected for each pair; for a given pair, each judge saw the same order.

Figure 2 includes the scores assigned to each of the justifications of entailment pair 75. Each pair of numbers in brackets is a score pair assigned by one judge. The first number in the pair is the understandability score and the second the correctness score. The correctness score is omitted (‘-’) when the understandability score is 1 or 2 because no correctness score was assigned in that case. The scores from the different judges are given in the same order for each justification.

With 100 entailment pairs evaluated by each of 6 judges assigning 2 separate scores, each run had a total of 1200 numbers assigned to it. Figure 3 shows two views of these numbers: a histogram of the number of justifications in the run that were assigned a given score value summed over all judges and all test pairs, and the overall mean score for the run<sup>1</sup>. A correctness score that was not assigned because understandability was too poor is displayed as a score of 0 in the histogram and treated as a 0 in the computation of the mean. Understandability scores are shown toward the left in the figure and correctness scores toward the right.

The mean scores for correctness are fairly low

<sup>1</sup>Using the mean treats a score as an interval variable rather than an ordinal variable, a questionable treatment given a 5-point scale. A mean is a convenient summary, however.

<p>Run A:</p> <p>There is a relationship between Zoopraxiscope and Muybridge in both the text and hypothesis. The term "invention" is morphologically similar to "invented".</p> <p>Scores: [4,4] [4,4] [5,4] [4,1] [5,4], [3,2]</p>
<p>Run B:</p> <p>The text mentions 'Muybridge'. We can infer that Muybridge is inventor. From the fact that Muybridge is inventor, we can infer that Muybridge invented. We can conclude that the Zoopraxiscope was invented by Muybridge.</p> <p>Scores:[4,3] [3,3] [5,4] [5,1] [5,3] [3,2]</p>
<p>Run C:</p> <p>1: The Hypothesis could be precisely matched with content in the Text, with allowance for polarity and embedded contexts.</p> <p>2: Hypothesis words match well with words in the Text.</p> <p>3: text adjunct "called" of "invention" dropped on aligned hyp word "invented"</p> <p>Scores: [3,3] [4,4] [4,4] [2,-] [1,-] [2,-]</p>
<p>Runs D and F:</p> <p>The words in the hypothesis are all found, with the approximately all the exact same syntactic dependencies, also in the text. Therefore, I concluded that the given hypothesis:</p> <p style="padding-left: 2em;">The Zoopraxiscope was invented by Muybridge.</p> <p>is entailed by the given text</p> <p style="padding-left: 2em;">Muybridge had earlier developed an invention he called the Zoopraxiscope.</p> <p>Scores: [3,3] [4,3] [4,3] [5,1] [4,3] [2,-]</p>
<p>Run H:</p> <p>Yes!</p> <p>I have general knowledge that:</p> <p>IF Y is developed by X THEN Y is manufactured by X</p> <p>Here: X = Muybridge, Y = the invention</p> <p>Thus, here:</p> <p style="padding-left: 2em;">We are told in T: the invention is developed by Muybridge</p> <p style="padding-left: 2em;">Thus it follows that: the invention is manufactured by Muybridge</p> <p>In addition, I know:</p> <p style="padding-left: 2em;">"manufacture" and "invent" mean roughly the same thing</p> <p>Hence: The Zoopraxiscope was invented by Muybridge.</p> <p>Scores: [2,-] [4,1] [3,3] [3,1] [2,-] [1,-]</p>

Figure 2: Justification for entailment pair 75 from each justification run. Brackets contain the pair of scores assigned to the justification by one of the six human judges; the first number in the pair is the understandability score and the second is the correctness score.

for all runs. Recall, however, that the ‘correctness’ score was actually interpreted as compellingness. There were many justifications that were strictly correct but not very informative, and they received low correctness scores. For example, the low correctness scores for the justification from run A in Figure 2 were given because those judges did not feel that the fact that “invention and inventor are morphologically similar” was enough of an explanation. Mean

correctness scores were also affected by understandability. Since an unassigned correctness score was treated as a zero when computing the mean, systems with low understandability scores must have lower correctness scores. Nonetheless, it is also true that systems reached the correct entailment decision by faulty reasoning uncomfortably often, as illustrated by the justification from run H in Figure 2.



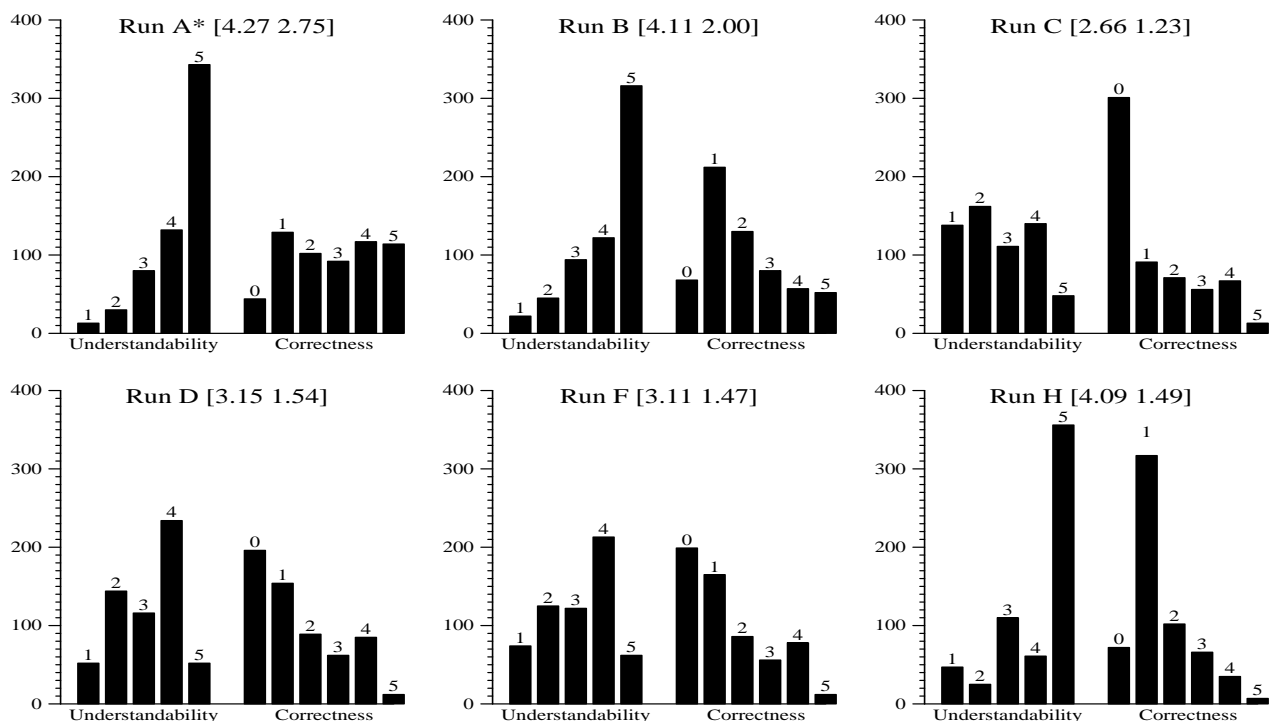


Figure 3: Number of justifications in a run that were assigned a particular score value summed over all judges and all test pairs. Brackets contain the overall mean understandability and correctness scores for the run. The starred run (A) is the manual run.

### 3.2 Human agreement

The most striking feature of the system results in Figure 3 is the variance in the scores. Not explicit in that figure, though illustrated in the example in Figure 2, is that different judges often gave widely different scores to the same justification. One systematic difference was immediately detected. The NIST judges have varying backgrounds with respect to mathematical training. Those with more training were more comfortable with, and often preferred, justifications expressed in mathematical notation; those with little training strongly disliked any mathematical notation in an explanation. This preference affected both the understandability and the correctness scores. Despite being asked to assign two separate scores, judges found it difficult to separate understandability and correctness. As a result, correctness scores were affected by presentation.

The scores assigned by different judges were sufficiently different to affect how runs compared to one another. This effect was quantified in the following way. For each entailment pair in the test set, the set of six runs was ranked by the scores assigned by

one assessor, with rank one assigned to the best run and rank six the worst run. If several systems had the same score, they were each assigned the mean rank for the tied set. (For example, if two systems had the same score that would rank them second and third, they were each assigned rank 2.5.) A run was then assigned its mean rank over the 100 justifications. Figure 4 shows how the mean rank of the runs varies by assessor. The x-axis in the figure shows the judge assigning the score and the y-axis the mean rank (remember that rank one is best). A run is plotted using its letter name consistent with previous figures, and lines connect the same system across different judges. Lines intersect demonstrating that different judges prefer different justifications.

After rating the 100 justifications, judges were asked to write a short summary of their impression of the task and what they looked for in a justification. These summaries did have some common themes. Judges prized conciseness and specificity, and expected (or at least hoped for) explanations in fluent English. Judges found “chatty” templates such as the one used in run H more annoying than engaging. Verbatim repetition of the text and hypothesis within

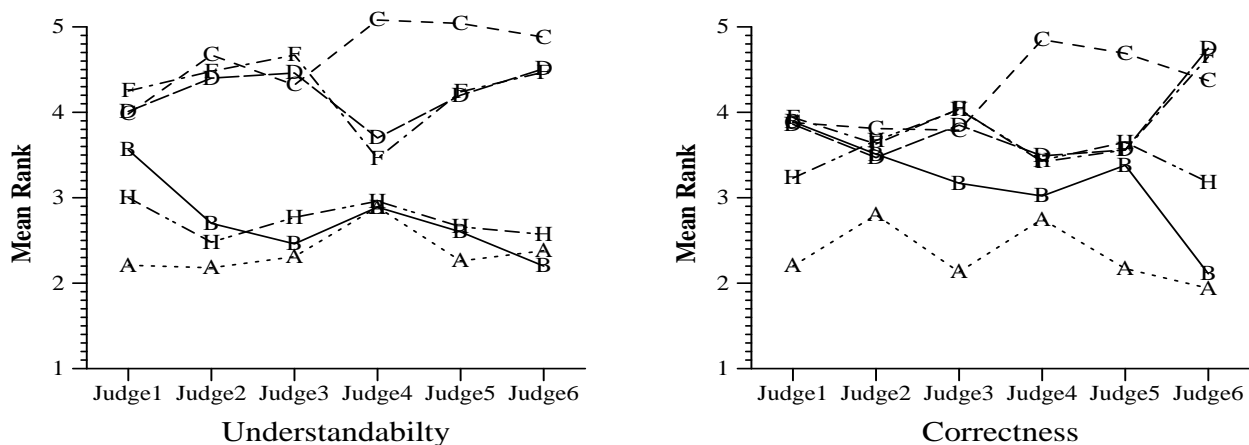


Figure 4: Relative effectiveness of runs as measured by mean rank.

the justification (as in runs D and F) was criticized as redundant. Generic phrases such as “there is a relation between” and “there is a match” were worse than useless: judges assigned no expository value to such assertions and penalized them as clutter.

Judges were also adverse to the use of system internals and jargon in the explanations. Some systems reported scores computed from WordNet (Fellbaum, 1998) or DIRT (Lin and Pantel, 2001). Such reports were penalized since the judges did not care what WordNet or DIRT are, and if they had cared, had no way to calibrate such a score. Similarly, linguistic jargon such as ‘polarity’ and ‘adjunct’ and ‘hyponym’ had little meaning for the judges.

Such qualitative feedback from the judges provides useful guidance to system builders on ways to explain system behavior. A broader conclusion from the justifications subtask is that it is premature for a quantitative evaluation of system-constructed explanations. The community needs a better understanding of the overall goal of justifications to develop a workable evaluation task. The relationships captured by many RTE entailment pairs are so obvious to humans (e.g., an inventor creates, a niece is a relative) that it is very unlikely end users would want explanations that include this level of detail. Having a true user task as a target would also provide needed direction as to the characteristics of those users, and thus allow judges to be more effective surrogates.

#### 4 Conclusion

The RTE-3 extended task provided an opportunity to examine systems’ abilities to detect contradiction and to provide explanations of their reasoning

when making entailment decisions. True contradiction was rare in the test set, accounting for approximately 10% of the test cases, though it is not possible to say whether this is a representative fraction for the text sources from which the test was drawn or simply a chance occurrence. Systems found detecting contradiction difficult, both missing it when it was present and finding it when it was not. Levels of human (dis)agreement regarding entailment and contradiction are such that test sets for a three-way decision task need to be substantially larger than for binary decisions for the evaluation to be both reliable and sensitive.

The justification task as implemented in RTE-3 is too abstract to make an effective evaluation task. Textual entailment decisions are at such a basic level of understanding for humans that human users don’t want explanations at this level of detail. User backgrounds have a profound effect on what presentation styles are acceptable in an explanation. The justification task needs to be more firmly situated in the context of a real user task so the requirements of the user task can inform the evaluation task.

#### Acknowledgements

The extended task of RTE-3 was supported by the Disruptive Technology Office (DTO) AQUAINT program. Thanks to fellow coordinators of the task, Chris Manning and Dan Moldovan, and to the participants for making the task possible.

## References

- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Lecture Notes in Computer Science*, volume 3944, pages 177–190. Springer-Verlag.
- Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.
- Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9. Association for Computational Linguistics.
- Dekang Lin and Patrick Pantel. 2001. DIRT —Discovery of inference rules from text. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 323–328.
- Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Proceedings of the Twenty-Third Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 200–207, July.
- Ellen M. Voorhees. 2000. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing and Management*, 36:697–716.

# Cohesive Phrase-based Decoding for Statistical Machine Translation

Colin Cherry\*

Microsoft Research

One Microsoft Way

Redmond, WA, 98052

colinc@microsoft.com

## Abstract

Phrase-based decoding produces state-of-the-art translations with no regard for syntax. We add syntax to this process with a cohesion constraint based on a dependency tree for the source sentence. The constraint allows the decoder to employ arbitrary, non-syntactic phrases, but ensures that those phrases are translated in an order that respects the source tree's structure. In this way, we target the phrasal decoder's weakness in order modeling, without affecting its strengths. To further increase flexibility, we incorporate cohesion as a decoder feature, creating a soft constraint. The resulting cohesive, phrase-based decoder is shown to produce translations that are preferred over non-cohesive output in both automatic and human evaluations.

## 1 Introduction

Statistical machine translation (SMT) is complicated by the fact that words can move during translation. If one assumes arbitrary movement is possible, that alone is sufficient to show the problem to be NP-complete (Knight, 1999). **Syntactic cohesion**<sup>1</sup> is the notion that all movement occurring during translation can be explained by permuting children in a parse tree (Fox, 2002). Equivalently, one can say that phrases in the source, defined by subtrees in its parse, remain contiguous after translation. Early

methods for syntactic SMT held to this assumption in its entirety (Wu, 1997; Yamada and Knight, 2001). These approaches were eventually superseded by tree transducers and tree substitution grammars, which allow translation events to span subtree units, providing several advantages, including the ability to selectively produce uncohesive translations (Eisner, 2003; Graehl and Knight, 2004; Quirk et al., 2005). What may have been forgotten during this transition is that there is a reason it was once believed that a cohesive translation model would work: for some language pairs, cohesion explains nearly all translation movement. Fox (2002) showed that cohesion is held in the vast majority of cases for English-French, while Cherry and Lin (2006) have shown it to be a strong feature for word alignment. We attempt to use this strong, but imperfect, characterization of movement to assist a non-syntactic translation method: phrase-based SMT.

Phrase-based decoding (Koehn et al., 2003) is a dominant formalism in statistical machine translation. Contiguous segments of the source are translated and placed in the target, which is constructed from left to right. The process iterates within a beam search until each word from the source has been covered by exactly one phrasal translation. Candidate translations are scored by a linear combination of models, weighted according to Minimum Error Rate Training or MERT (Och, 2003). Phrasal SMT draws strength from being able to memorize non-compositional and context-specific translations, as well as local reorderings. Its primary weakness is in movement modeling; its default distortion model applies a flat penalty to any deviation from source

\*Work conducted while at the University of Alberta.

<sup>1</sup>We use the term "syntactic cohesion" throughout this paper to mean what has previously been referred to as "phrasal cohesion", because the non-linguistic sense of "phrase" has become so common in machine translation literature.

order, forcing the decoder to rely heavily on its language model. Recently, a number of data-driven distortion models, based on lexical features and relative distance, have been proposed to compensate for this weakness (Tillman, 2004; Koehn et al., 2005; Al-Onaizan and Papineni, 2006; Kuhn et al., 2006).

There have been a number of proposals to incorporate syntactic information into phrasal decoding. Early experiments with syntactically-informed phrases (Koehn et al., 2003), and syntactic re-ranking of  $K$ -best lists (Och et al., 2004) produced mostly negative results. The most successful attempts at syntax-enhanced phrasal SMT have directly targeted movement modeling: Zens et al. (2004) modified a phrasal decoder with ITG constraints, while a number of researchers have employed syntax-driven source reordering before decoding begins (Xia and McCord, 2004; Collins et al., 2005; Wang et al., 2007).<sup>2</sup> We attempt something between these two approaches: our constraint is derived from a linguistic parse tree, but it is used inside the decoder, not as a preprocessing step.

We begin in Section 2 by defining syntactic cohesion so it can be applied to phrasal decoder output. Section 3 describes how to add both hard and soft cohesion constraints to a phrasal decoder. Section 4 provides our results from both automatic and human evaluations. Sections 5 and 6 provide a qualitative discussion of cohesive output and conclude.

## 2 Cohesive Phrasal Output

Previous approaches to measuring the cohesion of a sentence pair have worked with a word alignment (Fox, 2002; Lin and Cherry, 2003). This alignment is used to project the spans of subtrees from the source tree onto the target sentence. If a modifier and its head, or two modifiers of the same head, have overlapping spans in the projection, then this indicates a cohesion violation. To check phrasal translations for cohesion violations, we need a way to project the source tree onto the decoder’s output.

Fortunately, each phrase used to create the target sentence can be tracked back to its original source phrase, providing an alignment between source and

target phrases. Since each source token is used exactly once during translation, we can transform this phrasal alignment into a word-to-phrase alignment, where each source token is linked to a target phrase. We can then project the source subtree spans onto the target phrase sequence. Note that we never consider individual tokens on the target side, as their connection to the source tree is obscured by the phrasal abstraction that occurred during translation.

Let  $e_1^m$  be the input source sentence, and  $\bar{f}_1^p$  be the output target phrase sequence. Our word-to-phrase alignment  $a_i \in [1, p]$ ,  $1 \leq i \leq m$ , maps a source token position  $i$  to a target phrase position  $a_i$ . Next, we introduce our source dependency tree  $T$ . Each source token  $e_i$  is also a node in  $T$ . We define  $T(e_i)$  to be the subtree of  $T$  rooted at  $e_i$ . We define a local tree to be a head node and its immediate modifiers. With this notation in place, we can define our projected spans. Following Lin and Cherry (2003), we define a head span to be the projection of a single token  $e_i$  onto the target phrase sequence:

$$\text{span}H(e_i, T, a_1^m) = [a_i, a_i]$$

and the subtree span to be the projection of the subtree rooted at  $e_i$ :

$$\text{span}S(e_i, T, a_1^m) = \left[ \min_{\{j|e_j \in T(e_i)\}} a_j, \max_{\{k|e_k \in T(e_i)\}} a_k \right]$$

Consider the simple phrasal translation shown in Figure 1 along with a dependency tree for the English source. If we examine the local tree rooted at *likes*, we get the following projected spans:

$$\begin{aligned} \text{span}S(\textit{nobody}, T, a) &= [1, 1] \\ \text{span}H(\textit{likes}, T, a) &= [1, 1] \\ \text{span}S(\textit{pay}, T, a) &= [1, 2] \end{aligned}$$

For any local tree, we consider only the head span of the head, and the subtree spans of any modifiers.

Typically, cohesion would be determined by checking these projected spans for intersection. However, at this level of resolution, avoiding intersection becomes highly restrictive. The monotone translation in Figure 1 would become non-cohesive: *nobody* intersects with both its sibling *pay* and with its head *likes* at phrase index 1. This complication stems from the use of multi-word phrases that

<sup>2</sup>While certainly both syntactic and successful, we consider Hiero (Chiang, 2007) to be a distinct approach, and not an extension to phrasal decoding’s left-to-right beam search.

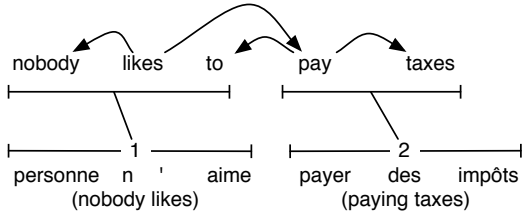


Figure 1: An English source tree with translated French output. Segments are indicated with underlined spans.

do not correspond to syntactic constituents. Restricting phrases to syntactic constituents has been shown to harm performance (Koehn et al., 2003), so we tighten our definition of a violation to disregard cases where the only point of overlap is obscured by our phrasal resolution. To do so, we replace span intersection with a new notion of span **innersection**.

Assume we have two spans  $[u, v]$  and  $[x, y]$  that have been sorted so that  $[u, v] \leq [x, y]$  lexicographically. We say that the two spans **innersect** if and only if  $x < v$ . So,  $[1, 3]$  and  $[2, 4]$  innersect, while  $[1, 3]$  and  $[3, 4]$  do not. One can think of innersection as intersection, minus the cases where the two spans share only a single boundary point, where  $x = v$ . When two projected spans innersect, it indicates that the second syntactic constituent must begin before the first ends. If the two spans in question correspond to nodes in the same local tree, innersection indicates an unambiguous cohesion violation. Under this definition, the translation in Figure 1 is cohesive, as none of its spans innersect.

Our hope is that syntactic cohesion will help the decoder make smarter distortion decisions. An example with distortion is shown in Figure 2. In this case, we present two candidate French translations of an English sentence, assuming there is no entry in the phrase table for “voting session.” Because the proper French construction is “session of voting”, the decoder has to move *voting* after *session* using a distortion operation. Figure 2 shows two methods to do so, each using an equal numbers of phrases. The projected spans for the local tree rooted at *begins* in each candidate are shown in Table 1. Note the innersection between the head *begins* and its modifier *session* in (b). Thus, a cohesion-aware system would receive extra guidance to select (a), which maintains the original meaning much better than (b).

Span	(a)	(b)
$spanS(session, T, a)$	[1,3]	[1,3]*
$spanH(begins, T, a)$	[4,4]	[2,2]*
$spanS(tomorrow, T, a)$	[4,4]	[4,4]

Table 1: Spans of the local trees rooted at *begins* from Figures 2 (a) and (b). Innersection is marked with a “\*”.

## 2.1 K-best List Filtering

A first attempt at using cohesion to improve SMT output would be to apply our definition as a filter on  $K$ -best lists. That is, we could have a phrasal decoder output a 1000-best list, and return the highest-ranked cohesive translation to the user. We tested this approach on our English-French development set, and saw no improvement in BLEU score. Error analysis revealed that only one third of the uncohesive translations had a cohesive alternative in their 1000-best lists. In order to reach the remaining two thirds, we need to constrain the decoder’s search space to explore only cohesive translations.

## 3 Cohesive Decoding

This section describes a modification to standard phrase-based decoding, so that the system is constrained to produce only cohesive output. This will take the form of a check performed each time a hypothesis is extended, similar to the ITG constraint for phrasal SMT (Zens et al., 2004). To create a such a check, we need to detect a cohesion violation inside a partial translation hypothesis. We cannot directly apply our span-based cohesion definition, because our word-to-phrase alignment is not yet complete. However, we can still detect violations, and we can do so before the spans involved are completely translated.

Recall that when two projected spans  $a$  and  $b$  ( $a < b$ ) innersect, it indicates that  $b$  begins before  $a$  ends. We can say that the translation of  $b$  interrupts the translation of  $a$ . We can enforce cohesion by ensuring that these **interruptions** never happen. Because the decoder builds its translations from left to right, eliminating interruptions amounts to enforcing the following rule: once the decoder begins translating any part of a source subtree, it must cover all

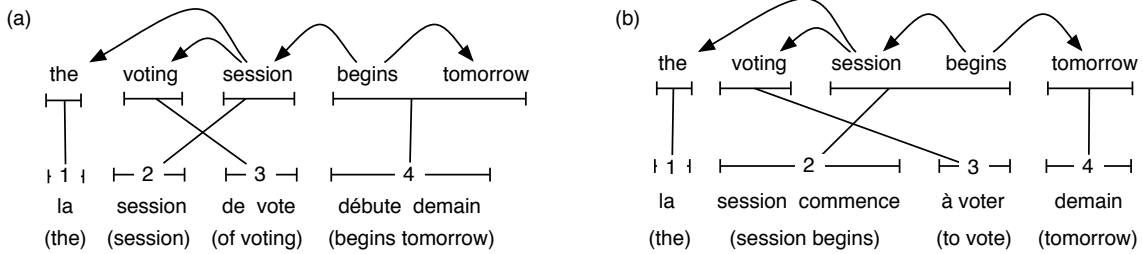


Figure 2: Two candidate translations for the same parsed source. (a) is cohesive, while (b) is not.

the words under that subtree before it can translate anything outside of it.

For example, in Figure 2b, the decoder translates *the*, which is part of  $T(\textit{session})$  in  $\bar{f}_1$ . In  $\bar{f}_2$ , it translates *begins*, which is outside  $T(\textit{session})$ . Since we have yet to cover *voting*, we know that the projected span of  $T(\textit{session})$  will end at some index  $v > 2$ , creating an innersection. This eliminates the hypothesis after having proposed only the first two phrases.

### 3.1 Algorithm

In this section, we formally define an interruption, and present an algorithm to detect one during decoding. During both discussions, we represent each target phrase as a set that contains the English tokens used in its translation:  $\bar{f}_j = \{e_i | a_i = j\}$ . Formally, an interruption occurs whenever the decoder would add a phrase  $\bar{f}_{h+1}$  to the hypothesis  $\bar{f}_1^h$ , and:

$$\begin{aligned}
 &\exists r \in T && \text{such that:} \\
 &\exists e \in T(r) && \text{s.t. } e \in \bar{f}_1^h && \text{(a. Started)} \\
 &\exists e' \notin T(r) && \text{s.t. } e' \in \bar{f}_{h+1} && \text{(b. Interrupted)} \\
 &\exists e'' \in T(r) && \text{s.t. } e'' \notin \bar{f}_1^{h+1} && \text{(c. Unfinished)}
 \end{aligned} \tag{1}$$

The key to checking for interruptions quickly is knowing which subtrees  $T(r)$  to check for qualities (1:a,b,c). A naïve approach would check every subtree that has begun translation in  $\bar{f}_1^h$ . Figure 3a highlights the roots of all such subtrees for a hypothetical  $T$  and  $\bar{f}_1^h$ . Fortunately, with a little analysis that accounts for  $\bar{f}_{h+1}$ , we can show that at most two subtrees need to be checked.

For a given interruption-free  $\bar{f}_1^h$ , we call subtrees that have begun translation, but are not yet complete, **open** subtrees. Only open subtrees can lead to interruptions. We can focus our interruption check on  $\bar{f}_h$ , the last phrase in  $\bar{f}_1^h$ , as any open subtree  $T(r)$  must contain at least one  $e \in \bar{f}_h$ . If this were not the

---

#### Algorithm 1 Interruption check.

---

- Get the left and right-most tokens used to create  $\bar{f}_h$ , call them  $e_L$  and  $e_R$
  - For each of  $e \in \{e_L, e_R\}$ :
    - i.  $r' \leftarrow e, r \leftarrow \textit{null}$   
 While  $\exists e' \in \bar{f}_{h+1}$  such that  $e' \notin T(r')$ :  
 $r \leftarrow r', r' \leftarrow \textit{parent}(r)$
    - ii. If  $r \neq \textit{null}$  and  $\exists e'' \in T(r)$  such that  $e'' \notin \bar{f}_1^{h+1}$ , then  $\bar{f}_{h+1}$  interrupts  $T(r)$ .
- 

case, then the open  $T(r)$  must have begun translation somewhere in  $\bar{f}_1^{h-1}$ , and  $T(r)$  would be interrupted by the placement of  $\bar{f}_h$ . Since our hypothesis  $\bar{f}_1^h$  is interruption-free, this is impossible. This leaves the subtrees highlighted in Figure 3b to be checked. Furthermore, we need only consider subtrees that contain the left and right-most source tokens  $e_L$  and  $e_R$  translated by  $\bar{f}_h$ . Since  $\bar{f}_h$  was created from a contiguous string of source tokens, any distinct subtree between these two endpoints will be completed within  $\bar{f}_h$ . Finally, for each of these focus points  $e_L$  and  $e_R$ , only the highest containing subtree  $T(r)$  that does not completely contain  $\bar{f}_{h+1}$  needs to be considered. Anything higher would contain all of  $\bar{f}_{h+1}$ , and would not satisfy requirement (1:b) of our interruption definition. Any lower subtree would be a descendant of  $r$ , and therefore the check for the lower subtree is subsumed by the check for  $T(r)$ . This leaves only two subtrees, highlighted in our running example in Figure 3c.

With this analysis in place, an extension  $\bar{f}_{h+1}$  of the hypothesis  $\bar{f}_1^h$  can be checked for interruptions with Algorithm 1. Step (i) in this algorithm finds an ancestor  $r'$  such that  $T(r')$  completely contains

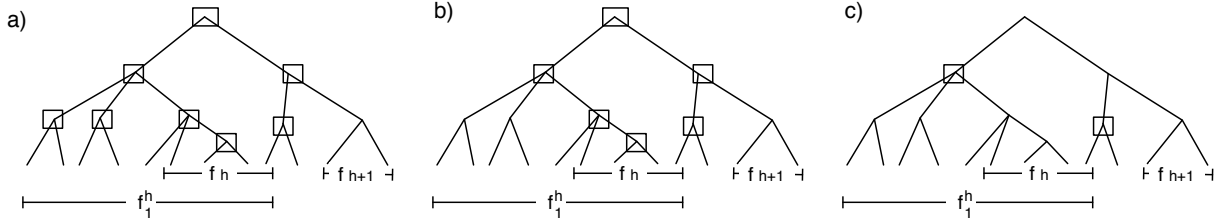


Figure 3: Narrowing down the source subtrees to be checked for completeness.

$\bar{f}_{h+1}$ , and then returns  $r$ , the highest node that **does not** contain  $\bar{f}_{h+1}$ . We know this  $r$  satisfies requirements (1:a,b). If there is no  $T(r)$  that does not contain  $\bar{f}_{h+1}$ , then  $e$  and its ancestors cannot lead to an interruption. Step (ii) then checks the coverage vector of the hypothesis<sup>3</sup> to make sure that  $T(r)$  is covered in  $\bar{f}_1^{h+1}$ . If  $T(r)$  is not complete in  $\bar{f}_1^{h+1}$ , then that satisfies requirement (1:c), which means an interruption has occurred.

For example, in Figure 2b, our first interruption occurs as we add  $\bar{f}_{h+1} = \bar{f}_2$  to  $\bar{f}_1^h = \bar{f}_1^1$ . The detection algorithm would first get the left and right boundaries of  $\bar{f}_1$ ; in this case, *the* is both  $e_L$  and  $e_R$ . Then, it would climb up the tree from *the* until it reached  $r' = \textit{begins}$  and  $r = \textit{session}$ . It would then check  $T(\textit{session})$  for coverage in  $\bar{f}_1^2$ . Since *voting*  $\in T(\textit{session})$  is not covered in  $\bar{f}_1^2$ , it would detect an interruption.

Walking up the tree takes at most linear time, and each check to see if  $T(r)$  contains all of  $\bar{f}_{h+1}$  can be performed in constant time, provided the source spans of each subtree have been precomputed. Checking to see if all of  $T(r)$  has been covered in Step (ii) takes at most linear time. This makes the entire process linear in the size of the source sentence.

### 3.2 Soft Constraint

Syntactic cohesion is not a perfect constraint for translation. Parse errors and systematic violations can create cases where cohesion works against the decoder. Fox (2002) demonstrated and counted cases where cohesion was not maintained in hand-aligned sentence-pairs, while Cherry and Lin (2006)

<sup>3</sup>This coverage vector is maintained by all phrasal decoders to track how much of the source sentence has been covered by the current partial translation, and to ensure that the same token is not translated twice.

showed that a soft cohesion constraint is superior to a hard constraint for word alignment. Therefore, we propose a soft version of our cohesion constraint. We perform our interruption check, but we do not invalidate any hypotheses. Instead, each hypothesis maintains a count of the number of extensions that have caused interruptions during its construction. This count becomes a feature in the decoder’s log-linear model, the weight of which is trained with MERT. After the first interruption, the exact meaning of further interruptions becomes difficult to interpret; but the interruption count does provide a useful estimate of the extent to which the translation is faithful to the source tree structure.

Initially, we were not certain to what extent this feature would be used by the MERT module, as BLEU is not always sensitive to syntactic improvements. However, trained with our French-English tuning set, the interruption count received the largest absolute feature weight, indicating, at the very least, that the feature is worth scaling to impact decoder.

### 3.3 Implementation

We modify the Moses decoder (Koehn et al., 2007) to translate head-annotated sentences. The decoder stores the flat sentence in the original sentence data structure, and the head-encoded dependency tree in an attached tree data structure. The tree structure caches the source spans corresponding to each of its subtrees. We then implement both a hard check for interruptions to be used before hypotheses are placed on the stack,<sup>4</sup> and a soft check that is used to calculate an interruption count feature.

<sup>4</sup>A hard cohesion constraint used in conjunction with a traditional distortion limit also requires a second linear-time check to ensure that all subtrees currently in progress can be finished under the constraints induced by the distortion limit.



Set	Cohesive	Uncohesive
Dev-Test	1170	330
Test	1563	437

Table 2: Number of sentences that receive cohesive translations from the baseline decoder. This property also defines our evaluation subsets.

## 4 Experiments

We have adapted the notion of syntactic cohesion so that it is applicable to phrase-based decoding. This results in a translation process that respects source-side syntactic boundaries when distorting phrases. In this section we will test the impact of such information on an English to French translation task.

### 4.1 Experimental Details

We test our cohesion-enhanced Moses decoder trained using 688K sentence pairs of Europarl French-English data, provided by the SMT 2006 Shared Task (Koehn and Monz, 2006). Word alignments are provided by GIZA++ (Och and Ney, 2003) with grow-diag-final combination, with infrastructure for alignment combination and phrase extraction provided by the shared task. We decode with Moses, using a stack size of 100, a beam threshold of 0.03 and a distortion limit of 4. Weights for the log-linear model are set using MERT, as implemented by Venugopal and Vogel (2005). Our tuning set is the first 500 sentences of the SMT06 development data. We hold out the remaining 1500 development sentences for development testing (dev-test), and the entirety of the provided 2000-sentence test set for blind testing (test). Since we require source dependency trees, all experiments test English to French translation. English dependency trees are provided by Minipar (Lin, 1994).

Our cohesion constraint directly targets sentences for which an unmodified phrasal decoder produces uncohesive output according to the definition in Section 2. Therefore, we present our results not only on each test set in its entirety, but also on the subsets defined by whether or not the baseline naturally produces a cohesive translation. The sizes of the resulting evaluation sets are given in Table 2.

Our development tests indicated that the soft and hard cohesion constraints performed somewhat sim-

ilarly, with the soft constraint providing more stable, and generally better results. We confirmed these trends on our test set, but to conserve space, we provide detailed results for only the soft constraint.

### 4.2 Automatic Evaluation

We first present our soft cohesion constraint’s effect on BLEU score (Papineni et al., 2002) for both our dev-test and test sets. We compare against an unmodified baseline decoder, as well as a decoder enhanced with a lexical reordering model (Tillman, 2004; Koehn et al., 2005). For each phrase pair in our translation table, the lexical reordering model tracks statistics on its reordering behavior as observed in our word-aligned training text. The lexical reordering model provides a good comparison point as a non-syntactic, and potentially orthogonal, improvement to phrase-based movement modeling. We use the implementation provided in Moses, with probabilities conditioned on bilingual phrases and predicting three orientation bins: straight, inverted and disjoint. Since adding features to the decoder’s log-linear model is straight-forward, we also experiment with a combined system that uses both the cohesion constraint and a lexical reordering model.

The results of our experiments are shown in Table 3, and reveal some interesting phenomena. First of all, looking across columns, we can see that there is a definite divide in BLEU score between our two evaluation subsets. Sentences with cohesive baseline translations receive much higher BLEU scores than those with uncohesive baseline translations. This indicates that the cohesive subset is easier to translate with a phrase-based system. Our definition of cohesive phrasal output appears to provide a useful feature for estimating translation confidence.

Comparing the baseline with and without the soft cohesion constraint, we see that cohesion has only a modest effect on BLEU, when measured on all sentence pairs, with improvements ranging between 0.2 and 0.5 absolute points. Recall that the majority of baseline translations are naturally cohesive. The cohesion constraint’s effect is much more pronounced on the more difficult uncohesive subsets, showing absolute improvements between 0.5 and 1.1 points.

Considering the lexical reordering model, we see that its effect is very similar to that of syntactic cohesion. Its BLEU scores are very similar, with lex-

System	Dev-Test			Test		
	All	Cohesive	Uncohesive	All	Cohesive	Uncohesive
base	32.04	33.80	27.46	32.35	33.78	28.73
lex	32.19	33.91	27.86	<b>32.71</b>	33.89	<b>29.66</b>
coh	32.22	33.82	<b>28.04</b>	<b>32.88</b>	<b>34.03</b>	<b>29.86</b>
lex+coh	<b>32.45</b>	34.12	<b>28.09</b>	<b>32.90</b>	34.04	<b>29.83</b>

Table 3: BLEU scores with an integrated soft cohesion constraint (coh) or a lexical reordering model (lex). Any system significantly better than base has been highlighted, as tested by bootstrap re-sampling with a 95% confidence interval.

ical reordering also affecting primarily the uncohesive subset. This similarity in behavior is interesting, as its data-driven, bilingual reordering probabilities are quite different from our cohesion flag, which is driven by monolingual syntax.

Examining the system that employs both movement models, we see that the combination (**lex+coh**) receives the highest score on the dev-test set. A large portion of the combined system’s gain is on the cohesive subset, indicating that the cohesion constraint may be enabling better use of the lexical reordering model on otherwise cohesive translations. Unfortunately, these same gains are not born out on the test set, where the lexical reordering model appears unable to improve upon the already strong performance of the cohesion constraint.

### 4.3 Human Evaluation

We also present a human evaluation designed to determine whether bilingual speakers prefer cohesive decoder output. Our comparison systems are the baseline decoder (**base**) and our soft cohesion constraint (**coh**). We evaluate on our dev-test set,<sup>5</sup> as it has our smallest observed BLEU-score gap, and we wish to determine if it is actually improving. Our experimental set-up is modeled after the human evaluation presented in (Collins et al., 2005). We provide two human annotators<sup>6</sup> a set of 75 English source sentences, along with a reference translation and a pair of translation candidates, one from each system. The annotators are asked to indicate which of the two system translations they prefer, or if they

<sup>5</sup>The cohesion constraint has no free parameters to optimize during development, so this does not create an advantage.

<sup>6</sup>Annotators were both native English speakers who speak French as a second language. Each has a strong comprehension of written French.

Annotator #1	Annotator #2			sum (#1)
	base	coh	equal	
base	<b>6</b>	7	1	14
coh	8	<b>35</b>	4	47
equal	7	4	<b>3</b>	14
sum (#2)	21	46	8	

Table 4: Confusion matrix from human evaluation.

consider them to be equal. To avoid bias, the competing systems were presented anonymously and in random order. Following (Collins et al., 2005), we provide the annotators with only short sentences: those with source sentences between 10 and 25 tokens long. Following (Callison-Burch et al., 2006), we conduct a targeted evaluation; we only draw our evaluation pairs from the uncohesive subset targeted by our constraint. All 75 sentences that meet these two criteria are included in the evaluation.

The aggregate results of our human evaluation are shown in the bottom row and right-most column of Table 4. Each annotator prefers **coh** in over 60% of the test sentences, and each prefers **base** in less than 30% of the test sentences. This presents strong evidence that we are having a consistent, positive effect on formerly non-cohesive translations. A complete confusion matrix indicating agreement between the two annotators is also given in Table 4. There are a few more off-diagonal points than one might expect, but it is clear that the two annotators are in agreement with respect to **coh**’s improvements. A combination annotator, which selects **base** or **coh** only when both human annotators agree and equal otherwise, finds **base** is preferred in only 8% of cases, compared to 47% for **coh**.

(1+)	creating structures that do not currently exist and reducing . . .
base	de créer des structures qui existent actuellement et ne pas réduire . . . <i>to create structures that <b>actually exist</b> and <b>do not reduce</b> . . .</i>
coh	de créer des structures qui n ’ existent pas encore et réduire . . . <i>to create structures that <b>do not yet exist</b> and <b>reduce</b> . . .</i>
(2−)	. . . repealed the 1998 directive banning advertising
base	. . . abrogée l’interdiction de la directive de 1998 de publicité <i>. . . <b>repealed the ban from the 1998 directive on advertising</b></i>
coh	. . . abrogée la directive de 1998 l’interdiction de publicité <i>. . . <b>repealed the 1998 directive the ban on advertising</b></i>

Table 5: A comparison of baseline and cohesion-constrained English-to-French translations, with English glosses.

## 5 Discussion

Examining the French translations produced by our cohesion constrained phrasal decoder, we can draw some qualitative generalizations. The constraint is used primarily to prevent distortion: it provides an intelligent estimate as to when source order must be respected. The resulting translations tend to be more literal than unconstrained translations. So long as the vocabulary present in our phrase table and language model supports a literal translation, cohesion tends to produce an improvement. Consider the first translation example shown in Table 5. In the baseline translation, the language model encourages the system to move the negation away from “exist” and toward “reduce.” The result is a tragic reversal of meaning in the translation. Our cohesion constraint removes this option, forcing the decoder to assemble the correct French construction for “does not yet exist.” The second example shows a case where our resources do not support a literal translation. In this case, we do not have a strong translation mapping to produce a French modifier equivalent to the English “banning.” Stuck with a noun form (“the ban”), the baseline is able to distort the sentence into something that is almost correct (the above gloss is quite generous). The cohesive system, even with a soft constraint, cannot reproduce the same movement, and returns a less grammatical translation.

We also examined cases where the decoder overrides the soft cohesion constraint and produces an uncohesive translation. We found this was done very rarely, and primarily to overcome parse errors. Only one correct syntactic construct repeatedly forced the

decoder to override cohesion: Minipar’s conjunction representation, which connects conjuncts in parent-child relationships, is at times too restrictive. A sibling representation, which would allow conjuncts to be permuted arbitrarily, may work better.

## 6 Conclusion

We have presented a definition of syntactic cohesion that is applicable to phrase-based SMT. We have used this definition to develop a linear-time algorithm to detect cohesion violations in partial decoder hypotheses. This algorithm was used to implement a soft cohesion constraint for the Moses decoder, based on a source-side dependency tree.

Our experiments have shown that roughly 1/5 of our baseline English-French translations contain cohesion violations, and these translations tend to receive lower BLEU scores. This suggests that cohesion could be a strong feature in estimating the confidence of phrase-based translations. Our soft constraint produced improvements ranging between 0.5 and 1.1 BLEU points on sentences for which the baseline produces uncohesive translations. A human evaluation showed that translations created using a soft cohesion constraint are preferred over uncohesive translations in the majority of cases.

**Acknowledgments** Special thanks to Dekang Lin, Shane Bergsma, and Jess Enright for their useful insights and discussions, and to the anonymous reviewers for their comments. The author was funded by Alberta Ingenuity and iCORE studentships.

## References

- Y. Al-Onaizan and K. Papineni. 2006. Distortion models for statistical machine translation. In *COLING-ACL*, pages 529–536, Sydney, Australia.
- C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *EACL*, pages 249–256.
- C. Cherry and D. Lin. 2006. Soft syntactic constraints for word alignment through discriminative training. In *COLING-ACL*, Sydney, Australia, July. Poster.
- D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, June.
- M. Collins, P. Koehn, and I. Kucerova. 2005. Clause restructuring for statistical machine translation. In *ACL*, pages 531–540.
- J. Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *ACL*, Sapporo, Japan. Short paper.
- H. J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *EMNLP*, pages 304–311.
- J. Graehl and K. Knight. 2004. Training tree transducers. In *HLT-NAACL*, pages 105–112, Boston, USA, May.
- K. Knight. 1999. Squibs and discussions: Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, December.
- P. Koehn and C. Monz. 2006. Manual and automatic evaluation of machine translation. In *HLT-NACCL Workshop on Statistical Machine Translation*, pages 102–121.
- P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*, pages 127–133.
- P. Koehn, A. Axelrod, A. Birch Mayne, C. Callison-Burch, M. Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *International Workshop on Spoken Language Translation*.
- P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*. Demonstration.
- R. Kuhn, D. Yuen, M. Simard, P. Paul, G. Foster, E. Joanis, and H. Johnson. 2006. Segment choice models: Feature-rich models for global distortion in statistical machine translation. In *HLT-NAACL*, pages 25–32, New York, NY.
- D. Lin and C. Cherry. 2003. Word alignment with cohesion constraint. In *HLT-NAACL*, pages 49–51, Edmonton, Canada, May. Short paper.
- D. Lin. 1994. Principar - an efficient, broad-coverage, principle-based parser. In *COLING*, pages 42–48, Kyoto, Japan.
- F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.
- F. J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 161–168.
- F. J. Och. 2003. Minimum error rate training for statistical machine translation. In *ACL*, pages 160–167.
- K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.
- C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *ACL*, pages 271–279, Ann Arbor, USA, June.
- C. Tillman. 2004. A unigram orientation model for statistical machine translation. In *HLT-NAACL*, pages 101–104. Short paper.
- A. Venugopal and S. Vogel. 2005. Considerations in maximum mutual information and minimum classification error training for statistical machine translation. In *EAMT*.
- C. Wang, M. Collins, and P. Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *EMNLP*, pages 737–745.
- D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.
- F. Xia and M. McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of Coling 2004*, pages 508–514.
- K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *ACL*, pages 523–530.
- R. Zens, H. Ney, T. Watanabe, and E. Sumita. 2004. Reordering constraints for phrase-based statistical machine translation. In *COLING*, pages 205–211, Geneva, Switzerland, August.

# Phrase Table Training For Precision and Recall: What Makes a Good Phrase and a Good Phrase Pair?

Yonggang Deng\* , Jia Xu<sup>+</sup> and Yuqing Gao\*

\*IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA

{ydeng, yuqing}@us.ibm.com

<sup>+</sup>Chair of Computer Science VI, RWTH Aachen University, D-52056 Aachen, Germany

xujia@cs.rwth-aachen.de

## Abstract

In this work, the problem of extracting phrase translation is formulated as an information retrieval process implemented with a log-linear model aiming for a balanced precision and recall. We present a generic phrase training algorithm which is parameterized with feature functions and can be optimized jointly with the translation engine to directly maximize the end-to-end system performance. Multiple data-driven feature functions are proposed to capture the quality and confidence of phrases and phrase pairs. Experimental results demonstrate consistent and significant improvement over the widely used method that is based on word alignment matrix only.

## 1 Introduction

Phrase has become the standard basic translation unit in Statistical Machine Translation (SMT) since it naturally captures context dependency and models internal word reordering. In a phrase-based SMT system, the phrase translation table is the defining component which specifies alternative translations and their probabilities for a given source phrase. In learning such a table from parallel corpus, two related issues need to be addressed (either separately or jointly): which pairs are considered valid translations and how to assign weights, such as probabilities, to them. The first problem is referred to as phrase pair extraction, which identifies phrase pairs that are supposed to be translations of each other. Methods have been proposed, based on syntax, that take advantage of linguistic constraints and alignment of grammatical structure, such as in Yamada

and Knight (2001) and Wu (1995). The most widely used approach derives phrase pairs from word alignment matrix (Och and Ney, 2003; Koehn et al., 2003). Other methods do not depend on word alignments only, such as directly modeling phrase alignment in a joint generative way (Marcu and Wong, 2002), pursuing information extraction perspective (Venugopal et al., 2003), or augmenting with model-based phrase pair posterior (Deng and Byrne, 2005).

Using relative frequency as translation probability is a common practice to measure goodness of a phrase pair. Since most phrases appear only a few times in training data, a phrase pair translation is also evaluated by lexical weights (Koehn et al., 2003) or term weighting (Zhao et al., 2004) as additional features to avoid overestimation. The translation probability can also be discriminatively trained such as in Tillmann and Zhang (2006).

The focus of this paper is the phrase pair extraction problem. As in information retrieval, precision and recall issues need to be addressed with a right balance for building a phrase translation table. High precision requires that identified translation candidates are accurate, while high recall wants as much valid phrase pairs as possible to be extracted, which is important and necessary for online translation that requires coverage. In the word-alignment derived phrase extraction approach, precision can be improved by filtering out most of the entries by using a statistical significance test (Johnson et al., 2007). On the other hand, there are valid translation pairs in the training corpus that are not learned due to word alignment errors as shown in Deng and Byrne (2005).

We would like to improve phrase translation accuracy and at the same time extract as many as possible valid phrase pairs that are missed due to incorrect word alignments. One approach is to leverage underlying word alignment quality such as in Ayan and Dorr (2006). In this work, we present a generic discriminative phrase pair extraction framework that can integrate multiple features aiming to identify correct phrase translation candidates. A significant deviation from most other approaches is that the framework is parameterized and can be optimized jointly with the decoder to maximize translation performance on a development set. Within the general framework, the main work is on investigating useful metrics. We employ features based on word alignment models and alignment matrix. We also propose information metrics that are derived from both bilingual and monolingual perspectives. All these features are data-driven and independent of languages. The proposed phrase extraction framework is general to apply linguistic features such as semantic, POS tags and syntactic dependency.

## 2 A Generic Phrase Training Procedure

Let  $\mathbf{e} = e_1^I$  denote an English sentence and  $\mathbf{f} = f_1^J$  denote its translation in a foreign language, say Chinese. Phrase extraction begins with sentence-aligned parallel corpora  $\{(\mathbf{e}_i, \mathbf{f}_i)\}$ . We use  $E = e_{i_b}^{i_e}$  and  $F = f_{j_b}^{j_e}$  to denote an English and foreign phrases respectively, where  $i_b(j_b)$  is the position in the sentence of the beginning word of the English(foreign) phrase and  $i_e(j_e)$  is the position of the ending word of the phrase.

We first train word alignment models and will use them to evaluate the goodness of a phrase and a phrase pair. Let  $f_k(E, F), k = 1, 2, \dots, K$  be  $K$  feature functions to be used to measure the quality of a given phrase pair  $(E, F)$ . The generic phrase extraction procedure is an evaluation, ranking, filtering, estimation and tuning process, presented in Algorithm 1.

Step 1 (line 1) is the preparation stage. Beginning with a flat lexicon, we train IBM Model-1 word alignment model with 10 iterations for each translation direction. We then train HMM word alignment models (Vogel et al., 1996) in two directions simultaneously by merging statistics collected in the

---

### Algorithm 1 A Generic Phrase Training Procedure

---

- 1: Train Model-1 and HMM word alignment models
  - 2: **for all** sentence pair  $(\mathbf{e}, \mathbf{f})$  **do**
  - 3:   Identify candidate phrases on each side
  - 4:   **for all** candidate phrase pair  $(E, F)$  **do**
  - 5:     Calculate its feature function values  $f_k$
  - 6:     Obtain the score  $q(E, F) = \sum_{k=1}^K \lambda_k f_k(E, F)$
  - 7:   **end for**
  - 8:   Sort candidate phrase pairs by their final scores  $q$
  - 9:   Find the maximum score  $qm = \max q(E, F)$
  - 10:   **for all** candidate phrase pair  $(E, F)$  **do**
  - 11:     If  $q(E, F) \geq qm - \tau$ , dump the pair into the pool
  - 12:   **end for**
  - 13: **end for**
  - 14: Built a phrase translation table from the phrase pair pool
  - 15: Discriminatively train feature weights  $\lambda_k$  and threshold  $\tau$
- 

E-step from two directions motivated by Zens et al. (2004) with 5 iterations. We use these models to define the feature functions of candidate phrase pairs such as phrase pair posterior distribution. More details will be given in Section 3.

Step 2 (line 2) consists of phrase pair evaluation, ranking and filtering. Usually all n-grams up to a pre-defined length limit are considered as candidate phrases. This is also the place where linguistic constraints can be applied, say to avoid non-compositional phrases (Lin, 1999). Each normalized feature score derived from word alignment models or language models will be log-linearly combined to generate the final score. Phrase pair filtering is simply thresholding on the final score by comparing to the maximum within the sentence pair. Note that under the log-linear model, applying threshold for filtering is equivalent to comparing the “likelihood” ratio.

Step 3 (line 14) pools all candidate phrase pairs that pass the threshold testing and estimates the final phrase translation table by maximum likelihood criterion. For each candidate phrase pair which is above the threshold, we assign HMM-based phrase pair posterior as its soft count when dumping them into the global phrase pair pool. Other possibilities for the weighting include assigning constant one or the exponential of the final score etc.

One of the advantages of the proposed phrase training algorithm is that it is a parameterized procedure that can be optimized jointly with the trans-

lation engine to minimize the final translation errors measured by automatic metrics such as BLEU (Papineni et al., 2002). In the final step 4 (line 15), parameters  $\{\lambda_k, \tau\}$  are discriminatively trained on a development set using the downhill simplex method (Nelder and Mead, 1965).

This phrase training procedure is general in the sense that it is configurable and trainable with different feature functions and their parameters. The commonly used phrase extraction approach based on word alignment heuristics (referred as *ViterbiExtract* algorithm for comparison in this paper) as described in (Och, 2002; Koehn et al., 2003) is a special case of the algorithm, where candidate phrase pairs are restricted to those that respect word alignment boundaries.

We rely on multiple feature functions that aim to describe the quality of candidate phrase translations and the generic procedure to figure out the best way of combining these features. A good feature function pops up valid translation pairs and pushes down incorrect ones.

### 3 Features

Now we present several feature functions that we investigated to help extracting correct phrase translations. All these features are data-driven and defined based on models, such as statistical word alignment model or language model.

#### 3.1 Model-based Phrase Pair Posterior

In a statistical generative word alignment model (Brown et al., 1993), it is assumed that (i) a random variable  $\mathbf{a}$  specifies how each target word  $f_j$  is generated by (therefore aligned to) a source<sup>1</sup> word  $e_{a_j}$ ; and (ii) the likelihood function  $f(\mathbf{f}, \mathbf{a}|\mathbf{e})$  specifies a generative procedure from the source sentence to the target sentence. Given a phrase pair in a sentence pair, there will be many generative paths that align the source phrase to the target phrase. The likelihood of those generative procedures can be accumulated to get the likelihood of the phrase pair (Deng and Byrne, 2005). This is implemented as the summation of the likelihood function over all valid hidden word alignments.

<sup>1</sup>The word *source* and *target* are in the sense of word alignment direction, not as in the source-channel formulation.

More specifically, let  $A_{(i_1, i_2)}^{(j_1, j_2)}$  be the set of word alignment  $\mathbf{a}$  that aligns the source phrase  $e_{i_1}^{j_1}$  to the target phrase  $f_{j_1}^{j_2}$  (links to NULL word are ignored for simplicity):

$$A_{(i_1, i_2)}^{(j_1, j_2)} = \{\mathbf{a} : a_j \in [i_1, i_2] \text{ iff } j \in [j_1, j_2]\}$$

The alignment set given a phrase pair ignores those pairs with word links across the phrase boundary. Consequently, the phrase-pair posterior distribution is defined as

$$P_{\theta}(e_{i_1}^{j_1} \rightarrow f_{j_1}^{j_2} | \mathbf{e}, \mathbf{f}) = \frac{\sum_{\mathbf{a} \in A_{(i_1, i_2)}^{(j_1, j_2)}} f(\mathbf{a}, \mathbf{f} | \mathbf{e}; \theta)}{\sum_{\mathbf{a}} f(\mathbf{a}, \mathbf{f} | \mathbf{e}; \theta)} \quad (1)$$

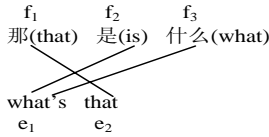
Switching the source and the target, we can obtain the posterior distribution in another translation direction. This distribution is applicable to all word alignment models that follow assumptions (i) and (ii). However, the complexity of the likelihood function could make it impractical to calculate the summations in Equation 1 unless an approximation is applied.

Several feature functions will be defined on top of the posterior distribution. One of them is based on HMM word alignment model. We use the geometric mean of posteriors in two translation directions as a symmetric metric for phrase pair quality evaluation function under HMM alignment models. Table 1 shows the phrase pair posterior matrix of the example.

Replacing the word alignment model with IBM Model-1 is another feature function that we added. IBM Model-1 is simple yet has been shown to be effective in many applications (Och et al., 2004). There is a close form solution to calculate the phrase pair posterior under Model-1. Moreover, word to word translation table under HMM is more concentrated than that under Model-1. Therefore, the posterior distribution evaluated by Model-1 is smoother and potentially it can alleviate the overestimation problem in HMM especially when training data size is small.

#### 3.2 Bilingual Information Metric

Trying to find phrase translations for any possible n-gram is not a good idea for two reasons. First, due to data sparsity and/or alignment model’s capability, there would exist n-grams that cannot be aligned



	$e_1^1$	$e_1^2$	$e_2^2$	$H_{BL}(f_{j_1}^{j_2})$
$f_1^1$	0.0006	0.012	0.89	0.08
$f_1^2$	0.0017	0.035	0.343	0.34
$f_1^3$	0.07	0.999	0.0004	0.24
$f_2^2$	0.03	0.0001	0.029	0.7
$f_2^3$	0.89	0.006	0.006	0.05
$f_3^3$	0.343	0.002	0.002	0.06
$H_{BL}(e_{i_1}^{i_2})$	0.869	0.26	0.70	

Table 1: Phrase pair posterior distribution for the example

well, for instance, n-grams that are part of a paraphrase translation or metaphorical expression. To give an example, the unigram ‘tomorrow’ in ‘the day after tomorrow’ whose Chinese translation is a single word ‘后天’. Extracting candidate translations for such kind of n-grams for the sake of improving coverage (recall) might hurt translation quality (precision). We will define a confidence metric to estimate how reliably the model can align an n-gram in one side to a phrase on the other side given a parallel sentence. Second, some n-grams themselves carry no linguistic meaning; their phrase translations can be misleading, for example non-compositional phrases (Lin, 1999). We will address this in section 3.3.

Given a sentence pair, the basic assumption is that if the HMM word alignment model can align an English phrase well to a foreign phrase, the posterior distribution of the English phrase generating all foreign phrases on the other side is significantly biased. For instance, the posterior of one foreign phrase is far larger than that of the others. We use the entropy of the posterior distribution as the confidence metric:

$$H_{BL}(e_{i_1}^{i_2} | \mathbf{e}, \mathbf{f}) = H(\hat{P}_{\theta_{HMM}}(e_{i_1}^{i_2} \rightarrow *)) \quad (2)$$

where  $H(P) = -\sum_x P(x) \log P(x)$  is the entropy of a distribution  $P(x)$ ,  $\hat{P}_{\theta_{HMM}}(e_{i_1}^{i_2} \rightarrow *)$  is the normalized probability (sum up to 1) of the posterior  $P_{\theta_{HMM}}(e_{i_1}^{i_2} \rightarrow *)$  as defined in Equation 1. Low entropy signals a high confidence that the English phrase can be aligned correctly. On the other hand, high entropy implies ambiguity presented in discriminating the correct foreign phrase from the others from the viewpoint of the model.

Similarly we calculate the confidence metric of aligning a foreign phrase correctly with the word alignment model in foreign to English direction. Table 1 shows the entropy of phrases. The unigram of foreign side  $f_2^2$  is unlikely to survive with such high ambiguity. Adding the entropy in two directions defines the bilingual information metric as another feature function, which describes the reliability of aligning each phrase correctly by the model. Note that we used HMM word alignment model to find the posterior distribution. Other models such as Model-1 can be applied in the same way. This feature function quantitatively captures the goodness of phrases. During phrase pair ranking, it can help to move upward phrases that can be aligned well and push downward phrases that are difficult for the model to find correct translations.

### 3.3 Monolingual Information Metric

Now we turn to monolingual resources to evaluate the quality of an n-gram being a good phrase. A phrase in a sentence is specified by its boundaries. We assume that the boundaries of a good phrase should be the “right” place to break. More generally, we want to quantify how effective a word boundary is as a phrase boundary. One would perform say NP-chunking or parsing to avoid splitting a linguistic constituent. We apply a language model (LM) to describe the predictive uncertainty (PU) between words in two directions.

Given a history  $w_1^{n-1}$ , a language model specifies a conditional distribution of the future word being predicted to follow the history. We can find the entropy of such pdf:  $H_{LM}(w_1^{n-1}) = H(P(\cdot | w_1^{n-1}))$ . So given a sentence  $w_1^N$ , the PU of the boundary between word  $w_i$  and  $w_{i+1}$  is established by two-way entropy sum using a forward and backward language model:  $PU(w_1^N, i) = H_{LMF}(w_1^i) + H_{LMB}(w_N^{i+1})$

We assume that the higher the predictive uncertainty is, the more likely the left or right part of the word boundary can be “cut-and-pasted” to form another reasonable sentence. So a good phrase is characterized with high PU values on the boundaries. For example, in ‘we want to have a table near the window’, the PU value of the point after ‘table’ is 0.61, higher than that between ‘near’ and ‘the’ 0.3, using trigram LMs.

With this, the feature function derived from



monolingual clue for a phrase pair can be defined as the product of  $PU$ s of the four word boundaries.

### 3.4 Word Alignments Induced Metric

The widely used ViterbiExtract algorithm relies on word alignment matrix and no-crossing-link assumption to extract phrase translation candidates. Practically it has been proved to work well. However, discarding correct phrase pairs due to incorrect word links leaves room for improving recall. This is especially true for not significantly large training corpora. Provided with a word alignment matrix, we define within phrase pair consistency ratio (WPPCR) as another feature function. WPPCR was used as one of the scores in (Venugopal et al., 2003) for phrase extraction. It is defined as the number of consistent word links associated with any words within the phrase pair divided by the number of all word links associated with any words within the phrase pair. An inconsistent link connects a word within the phrase pair to a word outside the phrase pair. For example, the WPPCR for  $(e_1^2, f_1^2)$  in Table 1 is  $2/3$ . As a special case, the ViterbiExtract algorithm extracts only phrase pairs with WPPCR is 1.

To further discriminate the pairs with higher WPPCR from those with lower ratio, we apply a Bi-Linear Transform (BLT) (Oppenheim and Schafer, 1989) mapping. BLT is commonly used in signal processing to attenuate the low frequency parts. When used to map WPPCR, it exaggerates the difference between phrase pairs with high WPPCR and those with low WPPCR, making the pairs with low ratio more unlikely to be selected as translation candidates. One of the nice properties of BLT is that there is a parameter that can be changed to adjust the degree of attenuation, which provides another dimension for system optimization.

## 4 Experimental Results

We evaluate the effect of the proposed phrase extraction algorithm with translation performance. We do experiments on IWSLT (Paul, 2006) 2006 Chinese-English corpus. The task is to translate Chinese utterances in travel domain into English. We report only text (speech transcription) translation results.

The training corpus consists of 40K Chinese-English parallel sentences in travel domain with to-

Eval Set	04dev	04test	05test	06dev	06test
# of sentences	506	500	506	489	500
# of words	2808	2906	3209	5214	5550
# of refs	16	16	16	7	7

Table 2: Dev/test set statistics

tal 306K English words and 295K Chinese words. In the data processing step, Chinese characters are segmented into words. English text are normalized and lowercased. All punctuation is removed.

There are five sets of evaluation sentences in tourism domain for development and test. Their statistics are shown in Table 2. We will tune training and decoding parameters on 06dev and report results on other sets.

### 4.1 Training and Translation Setup

Our decoder is a phrase-based multi-stack implementation of the log-linear model similar to Pharaoh (Koehn et al., 2003). Like other log-linear model based decoders, active features in our translation engine include translation models in two directions, lexicon weights in two directions, language model, lexicalized distortion models, sentence length penalty and other heuristics. These feature weights are tuned on the dev set to achieve optimal translation performance using downhill simplex method. The language model is a statistical trigram model estimated with Modified Kneser-Ney smoothing (Chen and Goodman, 1996) using only English sentences in the parallel training data.

Starting from the collection of parallel training sentences, we build word alignment models in two translation directions, from English to Chinese and from Chinese to English, and derive two sets of Viterbi alignments. By combining word alignments in two directions using heuristics (Och and Ney, 2003), a single set of static word alignments is then formed. Based on alignment models and word alignment matrices, we compare different approaches of building a phrase translation table and show the final translation results. We measure translation performance by the BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) scores with multiple translation references.

BLEU Scores					
Table	04dev	04test	05test	06dev	06test
HMM	0.367	0.407	0.473	0.200	0.190
Model-4	0.380	0.403	0.485	0.210	0.204
New	0.411	0.427	0.500	0.216	0.208

METEOR Scores					
Table	04dev	04test	05test	06dev	06test
HMM	0.532	0.586	0.675	0.482	0.471
Model-4	0.540	0.593	0.682	0.492	0.480
New	0.568	0.614	0.691	0.505	0.487

Table 3: Translation Results

## 4.2 Translation Results

Our baseline phrase table training method is the ViterbiExtract algorithm. All phrase pairs with respect to the word alignment boundary constraint are identified and pooled to build phrase translation tables with the Maximum Likelihood criterion. We prune phrase translation entries by their probabilities. The maximum number of words in Chinese and English phrases is set to 8 and 25 respectively for all conditions<sup>2</sup>. We perform online style phrase training, i.e., phrase extraction is not particular for any evaluation set.

Two different word alignment models are trained as the baseline, one is symmetric HMM word alignment model, the other is IBM Model-4 as implemented in the GIZA++ toolkit (Och and Ney, 2003). The translation results as measured by BLEU and METEOR scores are presented in Table 3. We notice that Model-4 based phrase table performs roughly 1% better in terms of both BLEU and METEOR scores than that based on HMM.

We follow the generic phrase training procedure as described in section 2. The most time consuming part is calculating posteriors, which is carried out in parallel with 30 jobs in less than 1.5 hours.

We use the Viterbi word alignments from HMM to define within phrase pair consistency ratio as discussed in section 3.4. Although Table 3 implies that Model-4 word alignment quality is better than that of HMM, we did not get benefits by switching to Model-4 to compute word alignments based feature values.

In estimating phrase translation probability, we use accumulated HMM-based phrase pair posteriors

<sup>2</sup>We chose large numbers for phrase length limit to build a strong baseline and to avoid impact of longer phrase length.

as their ‘soft’ frequencies and then the final translation probability is the relative frequency. HMM-based posterior was shown to be better than treating each occurrence as count one.

Once we have computed all feature values for all phrase pairs in the training corpus, we discriminatively train feature weights  $\lambda_k$ s and the threshold  $\tau$  using the downhill simplex method to maximize the BLEU score on 06dev set. Since the translation engine implements a log-linear model, the discriminative training of feature weights in the decoder should be embedded in the whole end-to-end system jointly with the discriminative phrase table training process. This is globally optimal but computationally demanding. As a compromise, we fix the decoder feature weights and put all efforts on optimizing phrase training parameters to find out the best phrase table.

The translation results with the discriminatively trained phrase table are shown as the row of ‘‘New’’ in Table 3. We observe that the new approach is consistently better than the baseline ViterbiExtract algorithm with either Model-4 or HMM word alignments on all sets. Roughly, it has 0.5% higher BLEU score on 2006 sets and 1.5% to 3% higher on other sets than Model-4 based ViterbiExtract method. Similar superior results are observed when measured with METEOR score.

## 5 Discussions

The generic phrase training algorithm follows an information retrieval perspective as in (Venugopal et al., 2003) but aims to improve both precision and recall with the trainable log-linear model. A clear advantage of the proposed approach over the widely used ViterbiExtract method is trainability. Under the general framework, one can put as many features as possible together under the log-linear model to evaluate the quality of a phrase and a phrase pair. The phrase table extracting procedure is trainable and can be optimized jointly with the translation engine.

Another advantage is flexibility, which is provided partially by the threshold  $\tau$ . As the figure 1 shows, when we increase the threshold by allowing more candidate phrase pair hypothesized as valid translation, we observe the phrase table size increases monotonically. On the other hand, we notice

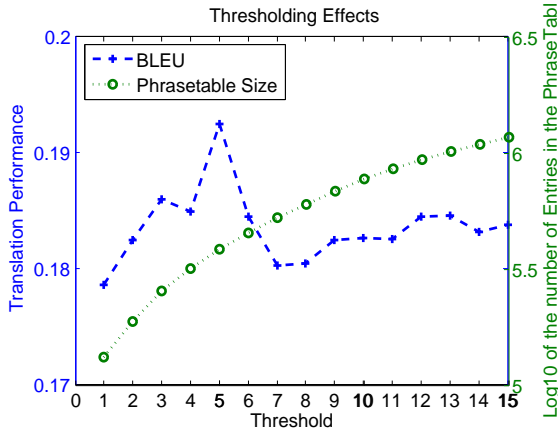


Figure 1: Thresholding effects on translation performance and phrase table size

that the translation performance improves gradually. After reaching its peak, the BLEU score drops as the threshold  $\tau$  increases. When  $\tau$  is large enough, the translation performance is not changing much but still worse than the peak value. It implies a balancing process between precision and recall. The final optimal threshold  $\tau$  is around 5.

The flexibility is also enabled by multiple configurable features used to evaluate the quality of a phrase and a phrase pair. Ideally, a perfect combination of feature functions divides the correct and incorrect candidate phrase pairs within a parallel sentence into two ordered separate sets. We use feature functions to decide the order and the threshold  $\tau$  to locate the boundary guided with a development set.

So the main issue to investigate now is which features are important and valuable in ranking candidate phrase pairs. We propose several information metrics derived from posterior distribution, language model and word alignments as feature functions. The ViterbiExtract is a special case where a single binary feature function defined from word alignments is used. Its good performance (as shown in Table 3) suggests that word alignments are very indicative of phrase pair quality. So we design comparative experiments to capture word alignment impact only. We start with basic features that include model-based posterior, bilingual and monolingual information metrics. Its results on different test sets are presented in the “basic” row of Table 4. We add word alignment feature (“+align” row), and

Features	04dev	04test	05test	06dev	06test
basic	0.393	0.406	0.496	0.205	0.199
+align	0.401	0.429	0.502	0.208	0.196
+align_BLT	0.411	0.427	0.500	0.216	0.208

Table 4: Translation Results (BLEU) of discriminative phrase training approach using different features



Features	04dev	04test	05test	06dev	06test
PP2	0.380	0.395	0.480	0.207	0.202
PP1+PP2	0.380	0.403	0.485	0.210	0.204
PP2+PP3	0.411	0.427	0.500	0.216	0.208
PP1+PP2+PP3	0.412	0.432	0.500	0.217	0.214

Table 5: Translation Results (BLEU) of Different Phrase Pair Combination

then apply bilinear transform to the consistency ratio WPPCR as described in section 3.4 (“+align\_BLT” row). The parameter controlling the degree of attenuation in BLT is also optimized together with other feature weights.

With the basic features, the new phrase extraction approach performs better than the baseline method with HMM word alignment models but similar to the baseline method with Model-4. With the word alignment based feature WPPCR, we obtain a 2% improvement on 04test set but not much on other sets except slight degradation on 06test. Finally, applying BLT transform to WPPCR leads to additional 0.8 BLEU point on 06dev set and 1.2 point on 06test set. This confirms the effectiveness of word alignment based features.

Now we compare the phrase table using the proposed method to that extracted using the baseline ViterbiExtract method with Model-4 word alignments. The Venn diagram in Table 5 shows how the two phrase tables overlap with each other and size of each part. As expected, they have a large number of common phrase pairs (PP2). The new method is able to extract more phrase pairs than the baseline with Model-4. PP1 is the set of phrase pairs found by Model-4 alignments. Removing PP1 from the baseline phrase table (comparing the first group of scores) or adding PP1 to the new phrase table

(the second group of scores) overall results in no or marginal performance change. On the other hand, adding phrase pairs extracted by the new method only (PP3) can lead to significant BLEU score increases (comparing row 1 vs. 3, and row 2 vs. 4).

## 6 Conclusions

In this paper, the problem of extracting phrase translation is formulated as an information retrieval process implemented with a log-linear model aiming for a balanced precision and recall. We have presented a generic phrase translation extraction procedure which is parameterized with feature functions. It can be optimized jointly with the translation engine to directly maximize the end-to-end translation performance. Multiple feature functions were investigated. Our experimental results on IWSLT Chinese-English corpus have demonstrated consistent and significant improvement over the widely used word alignment matrix based extraction method.<sup>3</sup>

**Acknowledgement** We would like to thank Xiaodong Cui, Radu Florian and other IBM colleagues for useful discussions and the anonymous reviewers for their constructive suggestions.

## References

- N. Ayan and B. Dorr. 2006. Going beyond AER: An extensive analysis of word alignments and their impact on MT. In *Proc. of ACL*, pages 9–16.
- S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.
- P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. 1993. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19:263–312.
- S. F. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. of ACL*, pages 310–318.
- Y. Deng and W. Byrne. 2005. HMM word and phrase alignment for statistical machine translation. In *Proc. of HLT-EMNLP*, pages 169–176.
- H. Johnson, J. Martin, G. Foster, and R. Kuhn. 2007. Improving translation quality by discarding most of the phrasetable. In *Proc. of EMNLP-CoNLL*, pages 967–975.
- P. Koehn, F. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 48–54.
- D. Lin. 1999. Automatic identification of non-compositional phrases. In *Proc. of ACL*, pages 317–324.
- D. Marcu and D. Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proc. of EMNLP*, pages 133–139.
- J. A. Nelder and R. Mead. 1965. A simplex method for function minimization. *Computer Journal*, 7:308–313.
- F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.
- F. J. Och, D. Gildea, and et al. 2004. A smorgasbord of features for statistical machine translation. In *Proc. of HLT-NAACL*, pages 161–168.
- F. Och. 2002. *Statistical Machine Translation: From Single Word Models to Alignment Templates*. Ph.D. thesis, RWTH Aachen, Germany.
- A. V. Oppenheim and R. W. Schaffer. 1989. *Discrete-Time Signal Processing*. Prentice-Hall.
- K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318.
- M. Paul. 2006. Overview of the IWSLT 2006 evaluation campaign. In *Proc. of IWSLT*, pages 1–15.
- C. Tillmann and T. Zhang. 2006. A discriminative global training algorithm for statistical MT. In *Proc. of ACL*, pages 721–728.
- A. Venugopal, S. Vogel, and A. Waibel. 2003. Effective phrase translation extraction from alignment models. In *Proc. of ACL*, pages 319–326.
- S. Vogel, H. Ney, and C. Tillmann. 1996. HMM based word alignment in statistical translation. In *Proc. of the COLING*.
- D. Wu. 1995. An algorithm for simultaneously bracketing parallel texts by aligning words. In *Proc. of ACL*, pages 244–251.
- K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL*, pages 523–530.
- R. Zens, E. Matusov, and H. Ney. 2004. Improved word alignment using a symmetric lexicon model. In *Proc. of COLING*, pages 36–42.
- B. Zhao, S. Vogel, M. Eck, and A. Waibel. 2004. Phrase pair rescoring with term weighting for statistical machine translation. In *Proc. of EMNLP*, pages 206–213.

<sup>3</sup>By parallelism, we have shown the feasibility and effectiveness (results not presented here) of the proposed method in handling millions of sentence pairs.

# Measure Word Generation for English-Chinese SMT Systems

Dongdong Zhang<sup>1</sup>, Mu Li<sup>1</sup>, Nan Duan<sup>2</sup>, Chi-Ho Li<sup>1</sup>, Ming Zhou<sup>1</sup>

<sup>1</sup>Microsoft Research Asia

Beijing, China

<sup>2</sup>Tianjin University

Tianjin, China

{dozhang, muli, v-naduan, chl, mingzhou}@microsoft.com

## Abstract

Measure words in Chinese are used to indicate the count of nouns. Conventional statistical machine translation (SMT) systems do not perform well on measure word generation due to data sparseness and the potential long distance dependency between measure words and their corresponding head words. In this paper, we propose a statistical model to generate appropriate measure words of nouns for an English-to-Chinese SMT system. We model the probability of measure word generation by utilizing lexical and syntactic knowledge from both source and target sentences. Our model works as a post-processing procedure over output of statistical machine translation systems, and can work with any SMT system. Experimental results show our method can achieve high precision and recall in measure word generation.

## 1 Introduction

In linguistics, *measure words* (MW) are words or morphemes used in combination with numerals or demonstrative pronouns to indicate the count of nouns<sup>1</sup>, which are often referred to as *head words* (HW).

Chinese measure words are grammatical units and occur quite often in real text. According to our survey on the measure word distribution in the Chinese Penn Treebank and the test datasets distributed by Linguistic Data Consortium (LDC) for Chinese-to-English machine translation evaluation, the average occurrence is 0.505 and 0.319 measure

words per sentence respectively. Unlike in Chinese, there is no special set of measure words in English. Measure words are usually used for mass nouns and any semantically appropriate nouns can function as the measure words. For example, in the phrase *three bottles of water*, the word *bottles* acts as a measure word. Countable nouns are almost never modified by measure words<sup>2</sup>. Numerals and indefinite articles are directly followed by countable nouns to denote the quantity of objects.

Therefore, in the English-to-Chinese machine translation task we need to take additional efforts to generate the missing measure words in Chinese. For example, when translating the English phrase *three books* into the Chinese phrases “三本书”, where *three* corresponds to the numeral “三” and *books* corresponds to the noun “书”, the Chinese measure word “本” should be generated between the numeral and the noun.

In most statistical machine translation (SMT) models (Och et al., 2004; Koehn et al., 2003; Chiang, 2005), some of measure words can be generated without modification or additional processing. For example, in above translation, the phrase translation table may suggest the word *three* be translated into “三”, “三本”, “三只”, etc, and the word *books* into “书”, “书本”, “名册” (scroll), etc. Then the SMT model selects the most likely combination “三本书” as the final translation result. In this example, a measure word candidate set consisting of “本” and “只” can be generated by bilingual phrases (or synchronous translation rules), and the best measure word “本” from the measure

<sup>1</sup> The uncommon cases of verbs are not considered.

<sup>2</sup> There are some exceptional cases, such as “100 head of cattle”. But they are very uncommon.

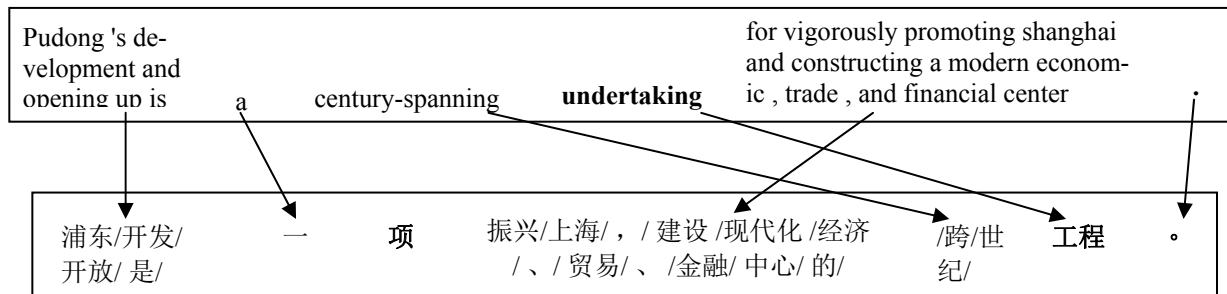


Figure 1. Example of long distance dependency between MW and its modified HW

word candidate set can be selected by the SMT decoder. However, as we will show below, existing SMT systems do not deal well with the measure word generation in general due to data sparseness and long distance dependencies between measure words and their corresponding head words.

Due to the limited size of bilingual corpora, many measure words, as well as the collocations between a measure and its head word, cannot be well covered by the phrase translation table in an SMT system. Moreover, Chinese measure words often have a long distance dependency to their head words which makes language model ineffective in selecting the correct measure words from the measure word candidate set. For example, in Figure 1 the distance between the measure word “项” and its head word “工程” (undertaking) is 15. In this case, an  $n$ -gram language model with  $n < 15$  cannot capture the MW-HW collocation. Table 1 shows the relative position’s distribution of head words around measure words in the Chinese Penn Treebank, where a negative position indicates that the head word is to the left of the measure word and a positive position indicates that the head word is to the right of the measure word. Although lots of measure words are close to the head words they modify, more than sixteen percent of measure words are far away from their corresponding head words (the absolute distance is more than 5).

To overcome the disadvantage of measure word generation in a general SMT system, this paper proposes a dedicated statistical model to generate measure words for English-to-Chinese translation.

We model the probability of measure word generation by utilizing rich lexical and syntactic knowledge from both source and target sentences. Three steps are involved in our method to generate measure words: Identifying the positions to gener-

ate measure words, collecting the measure word candidate set and selecting the best measure word. Our method is performed as a post-processing procedure of the output of SMT systems. The advantage is that it can be easily integrated into any SMT system. Experimental results show our method can significantly improve the quality of measure word generation. We also compared the performance of our model based on different contextual information, and show that both large-scale monolingual data and parallel bilingual data can be helpful to generate correct measure words.

Position	Occurrence	Position	Occurrence
1	39.5%	-1	0
2	15.7%	-2	0
3	4.7%	-3	8.7%
4	1.4%	-4	6.8%
5	2.1%	-5	4.3%
>5	8.8%	<-5	8.0%

Table 1. Position distribution of head words

## 2 Our Method

### 2.1 Measure word generation in Chinese

In Chinese, measure words are obligatory in certain contexts, and the choice of measure word usually depends on the head word’s semantics (e.g., shape or material). The set of Chinese measure words is a relatively close set and can be classified into two categories based on whether they have a corresponding English translation. Those not having an English counterpart need to be generated during translation. For those having English translations, such as “米” (meter), “吨” (ton), we just use the translation produced by the SMT system itself. According to our survey, about 70.4% of measure words in the Chinese Penn Treebank need

to be explicitly generated during the translation process.

In Chinese, there are generally stable linguistic collocations between measure words and their head words. Once the head word is determined, the collocated measure word can usually be selected accordingly. However, there is no easy way to identify head words in target Chinese sentences since for most of the time an SMT output is not a well formed sentence due to translation errors. Mistake of head word identification may cause low quality of measure word generation. In addition, sometimes the head word itself is not enough to determine the measure word. For example, in Chinese sentences “他家有 5 口人” (there are five people in his family) and “总共有 5 个人参加了会议” (a total of five people attended the meeting), where “人” (people) is the head word collocated with two different measure words “口” and “个”, we cannot determine the measure word just based on the head word “人”.

## 2.2 Framework

In our framework, a statistical model is used to generate measure words. The model is applied to SMT system outputs as a post-processing procedure. Given an English source sentence, an SMT decoder produces a target Chinese translation, in which positions for measure word generation are identified. Based on contextual information contained in both input source sentence and SMT system’s output translation, a measure word candidate set  $M$  is constructed. Then a measure word selection model is used to select the best one from  $M$ . Finally, the selected measure word is inserted into previously determined measure word slot in the SMT system’s output, yielding the final translation result.

## 2.3 Measure word position identification

To identify where to generate measure words in the SMT outputs, all positions after numerals are marked at first since measure words often follow numerals. For other cases in which measure words do not follow numerals (e.g., “许多/台/电脑” (many computers), where “台” is a measure word and “电脑” (computers) is its head word), we just mine the set of words which can be followed by measure words from training corpus. Most of

words in the set are pronouns such as “该” (this), “那” (that) and “若干” (several). In the SMT output, the positions after these words are also identified as candidate positions to generate measure words.

## 2.4 Candidate measure word generation

To avoid high computation cost, the measure word candidate set only consists of those measure words which can form valid MW-HW collocations with their head words. We assume that all the surrounding words within a certain window size centered on the given position to generate a measure word are potential head words, and require that a measure word candidate must collocate with at least one of the surrounding words. Valid MW-HW collocations are mined from the training corpus and a separate lexicon resource.

There is a possibility that the real head word is outside the window of given size. To address this problem, we also use a source window centered on the position  $p_s$ , which is aligned to the target measure word position  $p_t$ . The link between  $p_s$  and  $p_t$  can be inferred from SMT decoding result. Thus, the chance of capturing the best measure word increases with the aid of words located in the source window. For example, given the window size of 10, although the target head word “工程” (undertaking) in Figure 1 is located outside the target window, its corresponding source head word *undertaking* can be found in the source window. Based on this source head word, the best measure word “项” will be included into the candidate measure word set. This example shows how bilingual information can enrich the measure word candidate set.

Another special word {NULL} is always included in the measure word candidate set. {NULL} represents those measure words having a corresponding English translation as mentioned in Section 2.1. If {NULL} is selected, it means that we need not generate any measure word at the current position. Thus, no matter what kinds of measure words they are, we can handle the issue of measure word generation in a unified framework.

## 2.5 Measure word selection model

After obtaining the measure word candidate set  $M$ , a measure word selection model is employed to select the best one from  $M$ . Given the contextual information  $C$  in both source window and target

window, we model the measure word selection as finding the measure word  $m^*$  with highest posterior probability given  $C$ :

$$m^* = \operatorname{argmax}_{m \in M} P(m|C) \quad (1)$$

To leverage the collocation knowledge between measure words and head words, we extend (1) by introducing a hidden variable  $h$  where  $H$  represents all candidate head words located within the target window:

$$\begin{aligned} m^* &= \operatorname{argmax}_{m \in M} \sum_{h \in H} P(m, h|C) \\ &= \operatorname{argmax}_{m \in M} \sum_{h \in H} P(h|C)P(m|h, C) \end{aligned} \quad (2)$$

In (2),  $P(h|C)$  is the head word selection probability and is empirically estimated according to the position distribution of head words in Table 1.  $P(m|h, C)$  is the conditional probability of  $m$  given both  $h$  and  $C$ . We use maximum entropy model to compute  $P(m|h, C)$ :

$$P(m|h, C) = \frac{\exp(\sum_i \lambda_i f_i(m, C))}{\sum_{m' \in M} \exp(\sum_i \lambda_i f_i(m', C))} \quad (3)$$

Based on the different features used in the computation of  $P(m|h, C)$ , we can train two sub-models – a monolingual model (*Mo-ME*) which only uses monolingual (Chinese) features and a bilingual model (*Bi-ME*) which integrates bilingual features. The advantage of the Mo-ME model is that it can employ an unlimited monolingual target training corpora, while the Bi-ME model leverages rich features including both the source and target information and may improve the precision. Compared to the Mo-ME model, the Bi-ME model suffers from small scale of parallel training data. To leverage advantages of both models, we use a combined model *Co-ME*, by linearly combing the monolingual and bilingual sub-models:

$$m^* = \operatorname{argmax}_{m \in M} \lambda P_{Mo-ME} + (1 - \lambda) P_{Bi-ME}$$

where  $\lambda \in [0, 1]$  is a free parameter that can be optimized on held-out data and it was set to 0.39 in our experiments.

## 2.6 Features

The computation of Formula (3) involves the features listed in Table 2 where the Mo-ME model only employs target features and the Bi-ME model leverages both target features and source features.

For target features,  $n$ -gram language model score is defined as the sum of  $\log n$ -gram probabilities within the target window after the measure

word is filled into the measure word slot. The MW-HW collocation feature is defined to be a function  $f_1$  to capture the collocation between a measure word and a head word. For features of surrounding words, the feature function  $f_2$  is defined as 1 if a certain word exists at a certain position, otherwise 0. For example,  $f_2(\text{人}, -2) = 1$  means the second word on the left is “人”.  $f_2(\text{书}, 3) = 1$  means the third word on the right is “书”. For punctuation position feature function  $f_3$ , the feature value is 1 when there is a punctuation following the measure word, which indicates the target head word may appear to the left of measure word. Otherwise, it is 0. In practice, we can also ignore the position part, i.e., a word appears anywhere within the window is viewed as the same feature.

Target features	Source features
$n$ -gram language model score	MW-HW collocation
MW-HW collocation	surrounding words
surrounding words	source head word
punctuation position	POS tags

Table 2. Features used in our model

For source language side features, MW-HW collocation and surrounding words are used in a similar way as does with target features. The source head word feature is defined to be a function  $f_4$  to indicate whether a word  $e_i$  is the source head word in English according to a parse tree of the source sentence. Similar to the definition of lexical features, we also use a set of features based on POS tags of source language.

## 3 Model Training and Application

### 3.1 Training

We parsed English and Chinese sentences to get training samples for measure word generation model. Based on the source syntax parse tree, for each measure word, we identified its head word by using a toolkit from (Chiang and Bikel, 2002) which can heuristically identify head words for sub-trees. For the bilingual corpus, we also perform word alignment to get correspondences between source and target words. Then, the collocation between measure words and head words and their surrounding contextual information are extracted to train the measure word selection models. According to word alignment results, we classify



measure words into two classes based on whether they have non-null translations. We map Chinese measure words having non-null translations to a unified symbol {NULL} as mentioned in Section 2.4, indicating that we need not generate these kind of measure words since they can be translated from English.

In our work, the Berkeley parser (Petrov and Klein, 2007) was employed to extract syntactic knowledge from the training corpus. We ran GIZA++ (Och and Ney, 2000) on the training corpus in both directions with IBM model 4, and then applied the refinement rule described in (Koehn et al., 2003) to obtain a many-to-many word alignment for each sentence pair. We used the SRI Language Modeling Toolkit (Stolcke, 2002) to train a five-gram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998). The Maximum Entropy training toolkit from (Zhang, 2006) was employed to train the measure word selection model.

### 3.2 Measure word generation

As mentioned in previous sections, we apply our measure word generation module into SMT output as a post-processing step. Given a translation from an SMT system, we first determine the position  $p_t$  at which to generate a Chinese measure word. Centered on  $p_t$ , a surrounding word window with specified size is determined. From translation alignments, the corresponding source position  $p_s$  aligned to  $p_t$  can be referred. In the same way, a source window centered on  $p_s$  is determined as well. Then, contextual information within the windows in the source and the target sentence is extracted and fed to the measure word selection model. Meanwhile, the candidate set is obtained based on words in both windows. Finally, each measure word in the candidate set is inserted to the position  $p_t$ , and its score is calculated based on the models presented in Section 2.5. The measure word with the highest probability will be chosen.

There are two reasons why we perform measure word generation for SMT systems as a post-processing step. One is that in this way our method can be easily applied to any SMT system. The other is that we can leverage both source and target information during the measure word generation process. We do not integrate our measure word generation module into the SMT decoder since there is only little target contextual information available during SMT decoding. Moreover, as we

will show in experiment section, a pre-processing method does not work well when only source information is available.

## 4 Experiments

### 4.1 Data

In the experiments, the language model is a Chinese 5-gram language model trained with the Chinese part of the LDC parallel corpus and the Xinhua part of the Chinese Gigaword corpus with about 27 million words. We used an SMT system similar to Chiang (2005), in which FBIS corpus is used as the bilingual training data. The training corpus for Mo-ME model consists of the Chinese Peen Treebank and the Chinese part of the LDC parallel corpus with about 2 million sentences. The Bi-ME model is trained with FBIS corpus, whose size is smaller than that used in Mo-ME model training.

We extracted both development and test data set from years of NIST Chinese-to-English evaluation data by filtering out sentence pairs not containing measure words. The development set is extracted from NIST evaluation data from 2002 to 2004, and the test set consists of sentence pairs from NIST evaluation data from 2005 to 2006. There are 759 testing cases for measure word generation in our test data consisting of 2746 sentence pairs. We use the English sentences in the data sets as input to the SMT decoder, and apply our proposed method to generate measure words for the output from the decoder. Measure words in Chinese sentences of the development and test sets are used as references. When there are more than one measure words acceptable at some places, we manually augment the references with multiple acceptable measure words.

### 4.2 Baseline

Our baseline is the SMT output where measure words are generated by a Hiero-like SMT decoder as discussed in Section 1. Due to noises in the Chinese translations introduced by the SMT system, we cannot correctly identify all the positions to generate measure words. Therefore, besides precision we examine recall in our experiments.

### 4.3 Evaluation over SMT output

Table 3 and Table 4 show the precision and recall of our measure word generation method. From the

experimental results, the Mo-ME, Bi-ME and Co-ME models all outperform the baseline. Compared with the baseline, the Mo-ME method takes advantage of a large size monolingual training corpus and reduces the data sparseness problem. The advantage of the Bi-ME model is being able to make full use of rich knowledge from both source and target sentences. Also as shown in Table 3 and Table 4, the Co-ME model always achieve the best results when using the same window size since it leverages the advantage of both the Mo-ME and the Bi-ME models.

Wsize	Baseline	Mo-ME	Bi-ME	Co-ME
6	54.82%	64.29%	67.15%	67.66%
8		64.93%	68.50%	69.00%
10		64.72%	69.40%	69.58%
12		65.46%	69.40%	69.76%
14		65.61%	69.69%	70.03%

Table 3. Precision over SMT output

Wsize	Baseline	Mo-ME	Bi-ME	Co-ME
6	45.61%	51.48%	53.69%	54.09%
8		51.98%	54.75%	55.14%
10		51.81%	55.44%	55.58%
12		52.38%	55.44%	55.72%
14		52.50%	55.67%	55.93%

Table 4. Recall over SMT output

We can see that the Bi-ME model can achieve better results than the Mo-ME model in both recall and precision metrics although only a small sized bilingual corpus is used for Bi-ME model training. The reason is that the Mo-ME model cannot correctly handle the cases where head words are located outside the target window. However, due to word order differences between English and Chinese, when target head words are outside the target window, their corresponding source head words might be within the source window. The capacity of capturing head words is improved when both source and target windows are used, which demonstrates that bilingual knowledge is useful for measure word generation.

We compare the results for each model with different window sizes. Larger window size can lead to better results as shown in Table 3 and Table 4 since more contextual knowledge is used to model measure word generation. However, enlarging the window size does not bring significant improvements, The major reason is that even a small win-

dow size is already able to cover most of measure word collocations, as indicated by the position distribution of head words in Table 1.

The quality of the SMT output also affects the quality of measure word generation since our method is performed in a post-processing step over the SMT output. Although translation errors degrade the measure word generation accuracy, we achieve about 15% improvement in precision and a 10% increase in recall over baseline. We notice that the recall is relatively lower. Part of the reason is some positions to generate measure words are not successfully identified due to translation errors. In addition to precision and recall, we also evaluate the Bleu score (Papineni et al., 2002) changes before and after applying our measure word generation method to the SMT output. For our test data, we only consider sentences containing measure words for Bleu score evaluation. Our measure word generation step leads to a Bleu score improvement of 0.32 where the window size is set to 10, which shows that it can improve the translation quality of an English-to-Chinese SMT system.

#### 4.4 Evaluation over reference data

To isolate the impact of the translation errors in SMT output on the performance of our measure word generation model, we conducted another experiment with reference bilingual sentences in which measure words in Chinese sentences are manually removed. This experiment can show the performance upper bound of our method without interference from an SMT system. Table 5 shows the results. Compared to the results in Table 3, the precision improvement in the Mo-ME model is larger than that in the Bi-ME model, which shows that noisy translation of the SMT system has more serious influence on the Mo-ME model than the Bi-ME model. This also indicates that source information without noises is helpful for measure word generation.

Wsize	Mo-ME	Bi-ME	Co-ME
6	71.63%	74.92%	75.72%
8	73.80%	75.48%	76.20%
10	73.80%	74.76%	75.48%
12	73.80%	75.24%	75.96%
14	73.56%	75.48%	76.44%

Table 5. Results over reference data

## 4.5 Impacts of features

In this section, we examine the contribution of both target language based features and source language based features in our model. Table 6 and Table 7 show the precision and recall when using different features. The window size is set to 10. In the tables, *Lm* denotes the *n*-gram language model feature, *Tmh* denotes the feature of collocation between target head words and the candidate measure word, *Smh* denotes the feature of collocation between source head words and the candidate measure word, *Hs* denotes the feature of source head word selection, *Punc* denotes the feature of target punctuation position, *Tlex* denotes surrounding word features in translation, *Slex* denotes surrounding word features in source sentence, and *Pos* denotes Part-Of-Speech feature.

Feature setting	Precision	Recall
Baseline	54.82%	45.61%
<i>Lm</i>	51.11%	41.24%
+ <i>Tmh</i>	61.43%	49.22%
+ <i>Punc</i>	62.54%	50.08%
+ <i>Tlex</i>	64.80%	51.87%

Table 6. Feature contribution in Mo-ME model

Feature setting	Precision	Recall
Baseline	54.82%	45.61%
<i>Lm</i>	51.11%	41.24%
+ <i>Tmh</i> + <i>Smh</i>	64.50%	51.64%
+ <i>Hs</i>	65.32%	52.26%
+ <i>Punc</i>	66.29%	53.10%
+ <i>Pos</i>	66.53%	53.25%
+ <i>Tlex</i>	67.50%	54.02%
+ <i>Slex</i>	69.52%	55.54%

Table 7. Feature contribution in Bi-ME model

The experimental results show that all the features can bring incremental improvements. The method with only *Lm* feature performs worse than the baseline. However, with more features integrated, our method outperforms the baseline, which indicates each kind of features we selected is useful for measure word generation. According to the results, the feature of MW-HW collocation has much contribution to reducing the selection error of measure words given head words. The contribution of *Slex* feature explains that other surrounding words in source sentence are also helpful since head word determination in source language

might be incorrect due to errors in English parse trees. Meanwhile, the contribution from *Smh*, *Hs* and *Slex* features demonstrates that bilingual knowledge can play an important role for measure word generation. Compared with lexicalized features, we do not get much benefit from the *Pos* features.

## 4.6 Error analysis

We conducted an error analysis on 100 randomly selected sentences from the test data. There are four major kinds of errors as listed in Table 8. Most errors are caused by failures in finding positions to generate measure words. The main reason for this is some hint information used to identify measure word positions is missing in the noisy output of SMT systems. Two kinds of errors are introduced by incomplete head word and MW-HW collocation coverage, which can be solved by enlarging the size of training corpus. There are also head word selection errors due to incorrect syntax parsing.

Error type	Ratio
unseen head word	32.14%
unseen MW-HW collocation	10.71%
missing MW position	39.29%
incorrect HW selection	10.71%
others	7.14%

Table 8. Error distribution

## 4.7 Comparison with other methods

In this section we compare our statistical methods with the pre-processing method and the rule-based methods for measure word generation in a translation task.

In pre-processing method, only source language information is available. Given a source sentence, the corresponding syntax parse tree  $T_s$  is first constructed with an English parser. Then the pre-processing method chooses the source head word  $h_s$  based on  $T_s$ . The candidate measure word with the highest probability collocated with  $h_s$  is selected as the best result, where the measure word candidate set corresponding to each head word is mined over a bilingual training corpus in advance. We achieved precision 58.62% and recall 49.25%, which are worse than the results of our post-processing based methods. The weakness of the pre-processing method is twofold. One problem is data sparseness with respect to collocations be-

tween English head words and Chinese measure words. The other problem comes from the English head word selection error introduced by using source parse trees.

We also compared our method with a well-known rule-based machine translation system – SYSTRAN<sup>3</sup>. We translated our test data with SYSTRAN’s English-to-Chinese translation engine. The precision and recall are 63.82% and 51.09% respectively, which are also lower than our method.

## 5 Related Work

Most existing rule-based English-to-Chinese MT systems have a dedicated module handling measure word generation. In general a rule-based method uses manually constructed rule patterns to predict measure words. Like most rule based approaches, this kind of system requires lots of human efforts of experienced linguists and usually cannot easily be adapted to a new domain. The most relevant work based on statistical methods to our research might be statistical technologies employed to model issues such as morphology generation (Minkov et al., 2007).

## 6 Conclusion and Future Work

In this paper we propose a statistical model for measure word generation for English-to-Chinese SMT systems, in which contextual knowledge from both source and target sentences is involved. Experimental results show that our method not only achieves high precision and recall for generating measure words, but also improves the quality of English-to-Chinese SMT systems.

In the future, we plan to investigate more features and enlarge coverage to improve the quality of measure word generation, especially reduce the errors found in our experiments.

## Acknowledgements

Special thanks to David Chiang, Stephan Stiller and the anonymous reviewers for their feedback and insightful comments.

## References

Stanley F. Chen and Joshua Goodman. 1998. An Empirical study of smoothing techniques for language

modeling. *Technical Report TR-10-98, Harvard University Center for Research in Computing Technology*, 1998.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. *Proceedings of COLING '02*, 2002.

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

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *Proceedings of 45th Annual Meeting of the ACL*, pages 128–135.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of 38th Annual Meeting of the ACL*, pages 440–447.

Franz J. Och and Hermann Ney. 2004. *The alignment template approach to statistical machine translation*. *Computational Linguistics*, 30:417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the ACL*, pages 311–318.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL, 2007*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Le Zhang. MaxEnt toolkit. 2006. [http://homepages.inf.ed.ac.uk/s0450736/maxent\\_toolkit.html](http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html)

---

<sup>3</sup> <http://www.systransoft.com/>

# Bayesian Learning of Non-compositional Phrases with Synchronous Parsing

**Hao Zhang**

Computer Science Department  
University of Rochester  
Rochester, NY 14627  
zhanghao@cs.rochester.edu

**Chris Quirk**

Microsoft Research  
One Microsoft Way  
Redmond, WA 98052 USA  
chrisq@microsoft.com

**Robert C. Moore**

Microsoft Research  
One Microsoft Way  
Redmond, WA 98052 USA  
bobmoore@microsoft.com

**Daniel Gildea**

Computer Science Department  
University of Rochester  
Rochester, NY 14627  
gildea@cs.rochester.edu

## Abstract

We combine the strengths of Bayesian modeling and synchronous grammar in unsupervised learning of basic translation phrase pairs. The structured space of a synchronous grammar is a natural fit for phrase pair probability estimation, though the search space can be prohibitively large. Therefore we explore efficient algorithms for pruning this space that lead to empirically effective results. Incorporating a sparse prior using Variational Bayes, biases the models toward generalizable, parsimonious parameter sets, leading to significant improvements in word alignment. This preference for sparse solutions together with effective pruning methods forms a phrase alignment regimen that produces better end-to-end translations than standard word alignment approaches.

## 1 Introduction

Most state-of-the-art statistical machine translation systems are based on large phrase tables extracted from parallel text using word-level alignments. These word-level alignments are most often obtained using Expectation Maximization on the conditional generative models of Brown et al. (1993) and Vogel et al. (1996). As these word-level alignment models restrict the word alignment complexity by requiring each target word to align to zero or one source words, results are improved by aligning both source-to-target as well as target-to-source,

then heuristically combining these alignments. Finally, the set of phrases consistent with the word alignments are extracted from every sentence pair; these form the basis of the decoding process. While this approach has been very successful, poor word-level alignments are nonetheless a common source of error in machine translation systems.

A natural solution to several of these issues is unite the word-level and phrase-level models into one learning procedure. Ideally, such a procedure would remedy the deficiencies of word-level alignment models, including the strong restrictions on the form of the alignment, and the strong independence assumption between words. Furthermore it would obviate the need for heuristic combination of word alignments. A unified procedure may also improve the identification of non-compositional phrasal translations, and the attachment decisions for unaligned words.

In this direction, Expectation Maximization at the phrase level was proposed by Marcu and Wong (2002), who, however, experienced two major difficulties: computational complexity and controlling overfitting. Computational complexity arises from the exponentially large number of decompositions of a sentence pair into phrase pairs; overfitting is a problem because as EM attempts to maximize the likelihood of its training data, it prefers to directly explain a sentence pair with a single phrase pair.

In this paper, we attempt to address these two issues in order to apply EM above the word level.

We attack computational complexity by adopting the polynomial-time Inversion Transduction Grammar framework, and by only learning small *non-compositional* phrases. We address the tendency of EM to overfit by using Bayesian methods, where sparse priors assign greater mass to parameter vectors with fewer non-zero values therefore favoring shorter, more frequent phrases. We test our model by extracting longer phrases from our model’s alignments using traditional phrase extraction, and find that a phrase table based on our system improves MT results over a phrase table extracted from traditional word-level alignments.

## 2 Phrasal Inversion Transduction Grammar

We use a phrasal extension of Inversion Transduction Grammar (Wu, 1997) as the generative framework. Our ITG has two nonterminals:  $X$  and  $C$ , where  $X$  represents compositional phrase pairs that can have recursive structures and  $C$  is the pre-terminal over terminal phrase pairs. There are three rules with  $X$  on the left-hand side:

$$\begin{aligned} X &\rightarrow [X X], \\ X &\rightarrow \langle X X \rangle, \\ X &\rightarrow C. \end{aligned}$$

The first two rules are the straight rule and inverted rule respectively. They split the left-hand side constituent which represents a phrase pair into two smaller phrase pairs on the right-hand side and order them according to one of the two possible permutations. The rewriting process continues until the third rule is invoked.  $C$  is our unique pre-terminal for generating terminal multi-word pairs:

$$C \rightarrow \mathbf{e}/\mathbf{f}.$$

We parameterize our probabilistic model in the manner of a PCFG: we associate a multinomial distribution with each nonterminal, where each outcome in this distribution corresponds to an expansion of that nonterminal. Specifically, we place one multinomial distribution  $\theta_X$  over the three expansions of the nonterminal  $X$ , and another multinomial distribution  $\theta_C$  over the expansions of  $C$ . Thus, the parameters in our model can be listed as

$$\theta_{\mathbf{X}} = (P_{\diamond}, P_{\square}, P_C),$$

where  $P_{\diamond}$  is for the inverted rule,  $P_{\square}$  for the straight rule,  $P_C$  for the third rule, satisfying  $P_{\diamond} + P_{\square} + P_C = 1$ , and

$$\theta_{\mathbf{C}} = (P(\mathbf{e}/\mathbf{f}), P(\mathbf{e}'/\mathbf{f}'), \dots),$$

where  $\sum_{\mathbf{e}/\mathbf{f}} P(\mathbf{e}/\mathbf{f}) = 1$  is a multinomial distribution over phrase pairs.

This is our model in a nutshell. We can train this model using a two-dimensional extension of the inside-outside algorithm on bilingual data, assuming every phrase pair that can appear as a leaf in a parse tree of the grammar a valid candidate. However, it is easy to show that the maximum likelihood training will lead to the saturated solution where  $P_C = 1$  — each sentence pair is generated by a single phrase spanning the whole sentence. From the computational point of view, the full EM algorithm runs in  $O(n^6)$  where  $n$  is the average length of the two input sentences, which is too slow in practice.

The key is to control the number of parameters, and therefore the size of the set of candidate phrases. We deal with this problem in two directions. First we change the objective function by incorporating a prior over the phrasal parameters. This has the effect of preferring parameter vectors in  $\theta_{\mathbf{C}}$  with fewer non-zero values. Our second approach was to constrain the search space using simpler alignment models, which has the further benefit of significantly speeding up training. First we train a lower level word alignment model, then we place hard constraints on the phrasal alignment space using confident word links from this simpler model. Combining the two approaches, we have a staged training procedure going from the simplest unconstrained word based model to a constrained Bayesian word-level ITG model, and finally proceeding to a constrained Bayesian phrasal model.

## 3 Variational Bayes for ITG

Goldwater and Griffiths (2007) and Johnson (2007) show that modifying an HMM to include a sparse prior over its parameters and using Bayesian estimation leads to improved accuracy for unsupervised part-of-speech tagging. In this section, we describe a Bayesian estimator for ITG: we select parameters that optimize the probability of the data given a prior. The traditional estimation method for word

alignment models is the EM algorithm (Brown et al., 1993) which iteratively updates parameters to maximize the likelihood of the data. The drawback of maximum likelihood is obvious for phrase-based models. If we do not put any constraint on the distribution of phrases, EM overfits the data by memorizing every sentence pair. A sparse prior over a multinomial distribution such as the distribution of phrase pairs may bias the estimator toward skewed distributions that generalize better. In the context of phrasal models, this means learning the more representative phrases in the space of all possible phrases.

The Dirichlet distribution, which is parameterized by a vector of real values often interpreted as pseudo-counts, is a natural choice for the prior, for two main reasons. First, the Dirichlet is *conjugate* to the multinomial distribution, meaning that if we select a Dirichlet prior and a multinomial likelihood function, the posterior distribution will again be a Dirichlet. This makes parameter estimation quite simple. Second, Dirichlet distributions with small, non-zero parameters place more probability mass on multinomials on the edges or faces of the probability simplex, distributions with fewer non-zero parameters. Starting from the model from Section 2, we propose the following Bayesian extension, where  $A \sim \text{Dir}(B)$  means the random variable  $A$  is distributed according to a Dirichlet with parameter  $B$ :

$$\begin{aligned} \theta_{\mathbf{X}} \mid \alpha_X &\sim \text{Dir}(\alpha_X), \\ \theta_C \mid \alpha_C &\sim \text{Dir}(\alpha_C), \\ \left. \begin{array}{c} [X \ X] \\ \langle X \ X \rangle \\ C \end{array} \right| X &\sim \text{Multi}(\theta_{\mathbf{X}}), \\ \mathbf{e}/\mathbf{f} \mid C &\sim \text{Multi}(\theta_C). \end{aligned}$$

The parameters  $\alpha_X$  and  $\alpha_C$  control the sparsity of the two distributions in our model. One is the distribution of the three possible branching choices. The other is the distribution of the phrase pairs.  $\alpha_C$  is crucial, since the multinomial it is controlling has a high dimension. By adjusting  $\alpha_C$  to a very small number, we hope to place more posterior mass on parsimonious solutions with fewer but more confident and general phrase pairs.

Having defined the Bayesian model, it remains to decide the inference procedure. We chose Variational Bayes, for its procedural similarity to EM and ease of implementation. Another potential option would be Gibbs sampling (or some other sampling technique). However, in experiments in unsupervised POS tag learning using HMM structured models, Johnson (2007) shows that VB is more effective than Gibbs sampling in approaching distributions that agree with the Zipf’s law, which is prominent in natural languages.

Kurihara and Sato (2006) describe VB for PCFGs, showing the only need is to change the M step of the EM algorithm. As in the case of maximum likelihood estimation, Bayesian estimation for ITGs is very similar to PCFGs, which follows due to the strong isomorphism between the two models. Specific to our ITG case, the M step becomes:

$$\begin{aligned} \tilde{P}_{\square}^{(l+1)} &= \frac{\exp(\psi(E(X \rightarrow [X \ X]) + \alpha_X))}{\exp(\psi(E(X) + s\alpha_X))}, \\ \tilde{P}_{\langle \rangle}^{(l+1)} &= \frac{\exp(\psi(E(X \rightarrow \langle X \ X \rangle) + \alpha_X))}{\exp(\psi(E(X) + s\alpha_X))}, \\ \tilde{P}_C^{(l+1)} &= \frac{\exp(\psi(E(X \rightarrow C) + \alpha_X))}{\exp(\psi(E(X) + s\alpha_X))}, \\ \tilde{P}^{(l+1)}(\mathbf{e}/\mathbf{f}) &= \frac{\exp(\psi(E(\mathbf{e}/\mathbf{f}) + \alpha_C))}{\exp(\psi(E(C) + m\alpha_C))}, \end{aligned}$$

where  $\psi$  is the *digamma function* (Beal, 2003),  $s = 3$  is the number of right-hand-sides for  $X$ , and  $m$  is the number of observed phrase pairs in the data. The sole difference between EM and VB with a sparse prior  $\alpha$  is that the raw fractional counts  $c$  are replaced by  $\exp(\psi(c + \alpha))$ , an operation that resembles smoothing. As pointed out by Johnson (2007), in effect this expression adds to  $c$  a small value that asymptotically approaches  $\alpha - 0.5$  as  $c$  approaches  $\infty$ , and 0 as  $c$  approaches 0. For small values of  $\alpha$  the net effect is the opposite of typical smoothing, since it tends to redistribute probably mass away from unlikely events onto more likely ones.

## 4 Bitext Pruning Strategy

ITG is slow mainly because it considers every pair of spans in two sentences as a possible chart element. In reality, the set of useful chart elements is much

smaller than the possible  $\text{script}O(n^4)$ , where  $n$  is the average sentence length. Pruning the span pairs (bitext cells) that can participate in a tree (either as terminals or non-terminals) serves to not only speed up ITG parsing, but also to provide a kind of initialization hint to the training procedures, encouraging it to focus on promising regions of the alignment space.

Given a bitext cell defined by the four boundary indices  $(i, j, l, m)$  as shown in Figure 1a, we prune based on a figure of merit  $V(i, j, l, m)$  approximating the utility of that cell in a full ITG parse. The figure of merit considers the Model 1 scores of not only the words inside a given cell, but also all the words not included in the source and target spans, as in Moore (2003) and Vogel (2005). Like Zhang and Gildea (2005), it is used to prune bitext cells rather than score phrases. The total score is the product of the Model 1 probabilities for each column; “inside” columns in the range  $[l, m]$  are scored according to the sum (or maximum) of Model 1 probabilities for  $[i, j]$ , and “outside” columns use the sum (or maximum) of all probabilities not in the range  $[i, j]$ .

Our pruning differs from Zhang and Gildea (2005) in two major ways. First, we perform pruning using both directions of the IBM Model 1 scores; instead of a single figure of merit  $V$ , we have two:  $V_F$  and  $V_B$ . Only those spans that pass the pruning threshold in both directions are kept. Second, we allow whole spans to be pruned. The figure of merit for a span is  $V_F(i, j) = \max_{l, m} V_F(i, j, l, m)$ . Only spans that are within some threshold of the unrestricted Model 1 scores  $V_F$  and  $V_B$  are kept:

$$\frac{V_F(i, j)}{V_F} \geq \tau_s \text{ and } \frac{V_B(l, m)}{V_B} \geq \tau_s.$$

Amongst those spans retained by this first threshold, we keep only those bitext cells satisfying both

$$\frac{V_F(i, j, l, m)}{V_F(i, j)} \geq \tau_b \text{ and } \frac{V_B(i, j, l, m)}{V_B(l, m)} \geq \tau_b.$$

#### 4.1 Fast Tic-tac-toe Pruning

The tic-tac-toe pruning algorithm (Zhang and Gildea, 2005) uses dynamic programming to compute the product of inside and outside scores for all cells in  $\mathcal{O}(n^4)$  time. However, even this can be slow for large values of  $n$ . Therefore we describe an

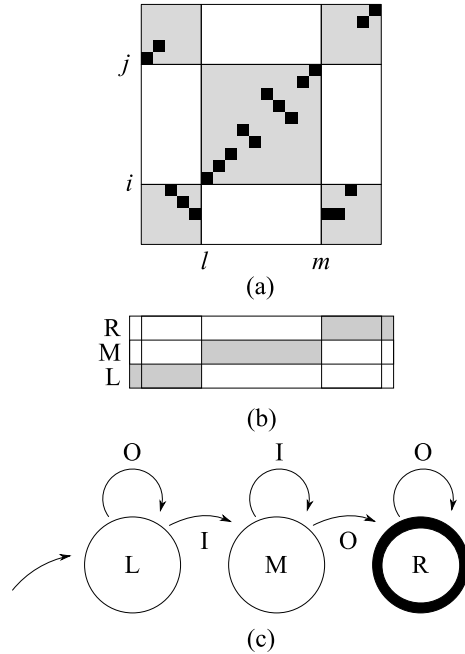


Figure 1: (a) shows the original tic-tac-toe score for a bitext cell  $(i, j, l, m)$ . (b) demonstrates the finite state representation using the machine in (c), assuming a fixed source span  $(i, j)$ .

improved algorithm with best case  $n^3$  performance. Although the worst case performance is also  $\mathcal{O}(n^4)$ , in practice it is significantly faster.

To begin, let us restrict our attention to the forward direction for a fixed source span  $(i, j)$ . Pruning bitext spans and cells requires  $V_F(i, j)$ , the score of the best bitext cell within a given span, as well as all cells within a given threshold of that best score. For a fixed  $i$  and  $j$ , we need to search over the starting and ending points  $l$  and  $m$  of the inside region. Note that there is an isomorphism between the set of spans and a simple finite state machine: any span  $(l, m)$  can be represented by a sequence of  $l$  OUTSIDE columns, followed by  $m - l + 1$  INSIDE columns, followed by  $n - m + 1$  OUTSIDE columns. This simple machine has the restricted form described in Figure 1c: it has three states,  $L$ ,  $M$ , and  $R$ ; each transition generates either an OUTSIDE column  $O$  or an INSIDE column  $I$ . The cost of generating an OUTSIDE at position  $a$  is  $O(a) = P(t_a|\text{NULL}) + \sum_{b \notin [i, j]} P(t_a|s_b)$ ; likewise the cost of generating an INSIDE column is  $I(a) = P(t_a|\text{NULL}) + \sum_{b \in [i, j]} P(t_a|s_b)$ , with



$O(0) = O(n + 1) = 1$  and  $I(0) = I(n + 1) = 0$ .

Directly computing  $O$  and  $I$  would take time  $\mathcal{O}(n^2)$  for each source span, leading to an overall runtime of  $\mathcal{O}(n^4)$ . Luckily there are faster ways to find the inside and outside scores. First we can pre-compute following arrays in  $\mathcal{O}(n^2)$  time and space:

$$\begin{aligned} \text{pre}[0, l] &:= P(t_l|\text{NULL}) \\ \text{pre}[i, l] &:= \text{pre}[i - 1, l] + P(t_l|s_i) \\ \text{suf}[n + 1, l] &:= 0 \\ \text{suf}[i, l] &:= \text{suf}[i + 1, l] + P(t_l|s_i) \end{aligned}$$

Then for any  $(i, j)$ ,  $O(a) = P(t_a|\text{NULL}) + \sum_{b \notin [i, j]} P(t_a|s_b) = \text{pre}[i - 1, a] + \text{suf}[j + 1, a]$ .  $I(a)$  can be incrementally updated as the source span varies: when  $i = j$ ,  $I(a) = P(t_a|\text{NULL}) + P(t_a|s_i)$ . As  $j$  is incremented, we add  $P(t_a|s_j)$  to  $I(a)$ . Thus we have linear time updates for  $O$  and  $I$ .

We can then find the best scoring sequence using the familiar Viterbi algorithm. Let  $\delta[a, \sigma]$  be the cost of the best scoring sequence ending in state  $\sigma$  at time  $a$ :

$$\begin{aligned} \delta[0, \sigma] &:= 1 \text{ if } \sigma = L; 0 \text{ otherwise} \\ \delta[a, L] &:= \delta[a - 1, L] \cdot O(a) \\ \delta[a, M] &:= \max_{\sigma \in L, M} \{\delta[a - 1, \sigma]\} \cdot I(a) \\ \delta[a, R] &:= \max_{\sigma \in M, R} \{\delta[a - 1, \sigma]\} \cdot O(a) \end{aligned}$$

Then  $V_F(i, j) = \delta[n + 1, R]$ , using the isomorphism between state sequences and spans. This linear time algorithm allows us to compute span pruning in  $\mathcal{O}(n^3)$  time. The same algorithm may be performed using the backward figure of merit after transposing rows and columns.

Having cast the problem in terms of finite state automata, we can use finite state algorithms for pruning. For instance, fixing a source span we can enumerate the target spans in decreasing order by score (Soong and Huang, 1991), stopping once we encounter the first span below threshold. In practice the overhead of maintaining the priority queue outweighs any benefit, as seen in Figure 2.

An alternate approach that avoids this overhead is to enumerate spans by position. Note that  $\delta[m, R] \cdot \prod_{a=m+1}^n O(a)$  is within threshold iff there is a span with right boundary  $m' < m$  within threshold. Furthermore if  $\delta[m, M] \cdot \prod_{a=m+1}^n O(a)$  is

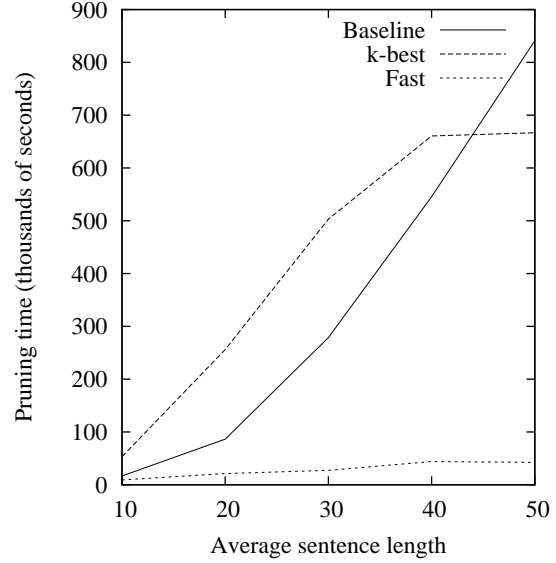


Figure 2: Speed comparison of the  $\mathcal{O}(n^4)$  tic-tac-toe pruning algorithm, the A\* top- $x$  algorithm, and the fast tic-tac-toe pruning. All produce the same set of bitext cells, those within threshold of the best bitext cell.

within threshold, then  $m$  is the right boundary within threshold. Using these facts, we can gradually sweep the right boundary  $m$  from  $n$  toward 1 until the first condition fails to hold. For each value where the second condition holds, we pause to search for the set of left boundaries within threshold.

Likewise for the left edge,  $\delta[l, M] \cdot \prod_{a=l+1}^m I(a) \cdot \prod_{a=m+1}^n O(a)$  is within threshold iff there is some  $l' < l$  identifying a span  $(l', m)$  within threshold. Finally if  $V(i, j, l, m) = \delta[l - 1, L] \cdot \prod_{a=l}^m I(a) \cdot \prod_{a=m+1}^n O(a)$  is within threshold, then  $(i, j, l, m)$  is a bitext cell within threshold. For right edges that are known to be within threshold, we can sweep the left edges leftward until the first condition no longer holds, keeping only those spans for which the second condition holds.

The filtering algorithm behaves extremely well. Although the worst case runtime is still  $\mathcal{O}(n^4)$ , the best case has improved to  $n^3$ ; empirically it seems to significantly reduce the amount of time spent exploring spans. Figure 2 compares the speed of the fast tic-tac-toe algorithm against the algorithm in Zhang and Gildea (2005).

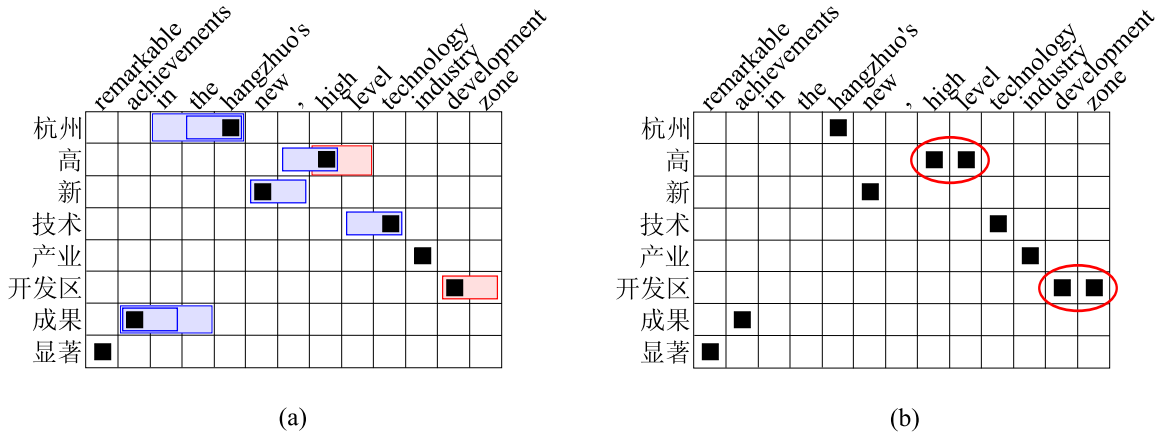


Figure 3: Example output from the ITG using non-compositional phrases. (a) is the Viterbi alignment from the word-based ITG. The shaded regions indicate phrasal alignments that are allowed by the non-compositional constraint; all other phrasal alignments will not be considered. (b) is the Viterbi alignment from the phrasal ITG, with the multi-word alignments highlighted.

## 5 Bootstrapping Phrasal ITG from Word-based ITG

This section introduces a technique that bootstraps candidate phrase pairs for phrase-based ITG from word-based ITG Viterbi alignments. The word-based ITG uses the same expansions for the non-terminal  $X$ , but the expansions of  $C$  are limited to generate only 1-1, 1-0, and 0-1 alignments:

$$\begin{aligned} C &\rightarrow e/f, \\ C &\rightarrow e/\epsilon, \\ C &\rightarrow \epsilon/f \end{aligned}$$

where  $\epsilon$  indicates that no word was generated. Broadly speaking, the goal of this section is the same as the previous section, namely, to limit the set of phrase pairs that needs to be considered in the training process. The tic-tac-toe pruning relies on IBM model 1 for scoring a given aligned area. In this part, we use word-based ITG alignments as anchor points in the alignment space to pin down the potential phrases. The scope of iterative phrasal ITG training, therefore, is limited to determining the boundaries of the phrases anchored on the given one-to-one word alignments.

The heuristic method is based on the Non-Compositional Constraint of Cherry and Lin (2007). Cherry and Lin (2007) use GIZA++ intersections which have high precision as anchor points in the

bitext space to constraint ITG phrases. We use ITG Viterbi alignments instead. The benefit is two-fold. First of all, we do not have to run a GIZA++ aligner. Second, we do not need to worry about non-ITG word alignments, such as the (2, 4, 1, 3) permutation patterns. GIZA++ does not limit the set of permutations allowed during translation, so it can produce permutations that are not reachable using an ITG.

Formally, given a word-based ITG alignment, the bootstrapping algorithm finds all the phrase pairs according to the definition of Och and Ney (2004) and Chiang (2005) with the additional constraint that each phrase pair contains at most one word link. Mathematically, let  $e(i, j)$  count the number of word links that are emitted from the substring  $e_{i\dots j}$ , and  $f(l, m)$  count the number of word links emitted from the substring  $f_{l\dots m}$ . The non-compositional phrase pairs satisfy

$$e(i, j) = f(l, m) \leq 1.$$

Figure 3 (a) shows all possible non-compositional phrases given the Viterbi word alignment of the example sentence pair.

## 6 Summary of the Pipeline

We summarize the pipeline of our system, demonstrating the interactions between the three main contributions of this paper: Variational Bayes, tic-tac-toe pruning, and word-to-phrase bootstrapping. We

start from sentence-aligned bilingual data and run IBM Model 1 in both directions to obtain two translation tables. Then we use the efficient bidirectional tic-tac-toe pruning to prune the bitext space within each of the sentence pairs; ITG parsing will be carried out on only this sparse set of bitext cells. The first stage of training is word-based ITG, using the standard iterative training procedure, except VB replaces EM to focus on a sparse prior. After several training iterations, we obtain the Viterbi alignments on the training data according to the final model. Now we transition into the second stage – the phrasal training. Before the training starts, we apply the non-compositional constraints over the pruned bitext space to further constrain the space of phrase pairs. Finally, we run phrasal ITG iterative training using VB for a certain number of iterations. In the end, a Viterbi pass for the phrasal ITG is executed to produce the non-compositional phrasal alignments. From this alignment, phrase pairs are extracted in the usual manner, and a phrase-based translation system is trained.

## 7 Experiments

The training data was a subset of 175K sentence pairs from the NIST Chinese-English training data, automatically selected to maximize character-level overlap with the source side of the test data. We put a length limit of 35 on both sides, producing a training set of 141K sentence pairs. 500 Chinese-English pairs from this set were manually aligned and used as a gold standard.

### 7.1 Word Alignment Evaluation

First, using evaluations of alignment quality, we demonstrate the effectiveness of VB over EM, and explore the effect of the prior.

Figure 4 examines the difference between EM and VB with varying sparse priors for the word-based model of ITG on the 500 sentence pairs, both after 10 iterations of training. Using EM, because of overfitting, AER drops first and increases again as the number of iterations varies from 1 to 10. The lowest AER using EM is achieved after the second iteration, which is .40. At iteration 10, AER for EM increases to .42. On the other hand, using VB, AER decreases monotonically over the 10 iterations and

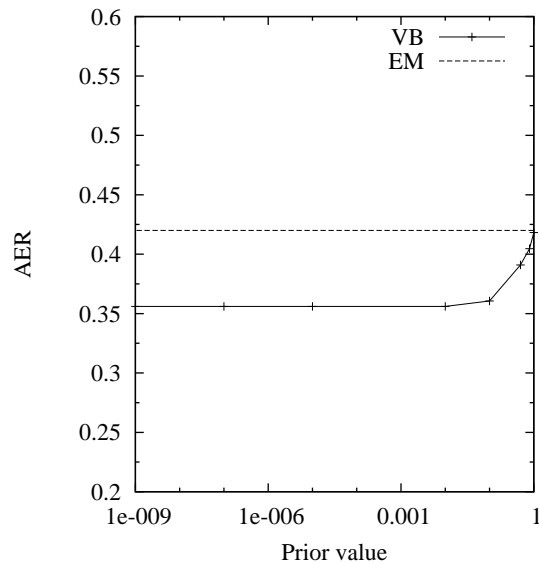


Figure 4: AER drops as  $\alpha_C$  approaches zero; a more sparse solution leads to better results.

stabilizes at iteration 10. When  $\alpha_C$  is  $1e - 9$ , VB gets AER close to .35 at iteration 10.

As we increase the bias toward sparsity, the AER decreases, following a long slow plateau. Although the magnitude of improvement is not large, the trend is encouraging.

These experiments also indicate that a very sparse prior is needed for machine translation tasks. Unlike Johnson (2007), who found optimal performance when  $\alpha$  was approximately  $10^{-4}$ , we observed monotonic increases in performance as  $\alpha$  dropped. The dimensionality of this MT problem is significantly larger than that of the sequence problem, though, therefore it may take a stronger push from the prior to achieve the desired result.

### 7.2 End-to-end Evaluation

Given an unlimited amount of time, we would tune the prior to maximize end-to-end performance, using an objective function such as BLEU. Unfortunately these experiments are very slow. Since we observed monotonic increases in alignment performance with smaller values of  $\alpha_C$ , we simply fixed the prior at a very small value ( $10^{-100}$ ) for all translation experiments. We do compare VB against EM in terms of final BLEU scores in the translation experiments to ensure that this sparse prior has a sig-

nificant impact on the output.

We also trained a baseline model with GIZA++ (Och and Ney, 2003) following a regimen of 5 iterations of Model 1, 5 iterations of HMM, and 5 iterations of Model 4. We computed Chinese-to-English and English-to-Chinese word translation tables using five iterations of Model 1. These values were used to perform tic-tac-toe pruning with  $\tau_b = 1 \times 10^{-3}$  and  $\tau_s = 1 \times 10^{-6}$ . Over the pruned charts, we ran 10 iterations of word-based ITG using EM or VB. The charts were then pruned further by applying the non-compositional constraint from the Viterbi alignment links of that model. Finally we ran 10 iterations of phrase-based ITG over the residual charts, using EM or VB, and extracted the Viterbi alignments.

For translation, we used the standard phrasal decoding approach, based on a re-implementation of the Pharaoh system (Koehn, 2004). The output of the word alignment systems (GIZA++ or ITG) were fed to a standard phrase extraction procedure that extracted all phrases of length up to 7 and estimated the conditional probabilities of source given target and target given source using relative frequencies. Thus our phrasal ITG learns only the minimal non-compositional phrases; the standard phrase-extraction algorithm learns larger combinations of these minimal units. In addition the phrases were annotated with lexical weights using the IBM Model 1 tables. The decoder also used a trigram language model trained on the target side of the training data, as well as word count, phrase count, and distortion penalty features. Minimum Error Rate training (Och, 2003) over BLEU was used to optimize the weights for each of these models over the development test data.

We used the NIST 2002 evaluation datasets for tuning and evaluation; the 10-reference development set was used for minimum error rate training, and the 4-reference test set was used for evaluation. We trained several phrasal translation systems, varying only the word alignment (or phrasal alignment) method.

Table 1 compares the four systems: the GIZA++ baseline, the ITG word-based model, the ITG multiword model using EM training, and the ITG multiword model using VB training. ITG-mwm-VB is our best model. We see an improvement of nearly

	<i>Development</i>	<i>Test</i>
GIZA++	37.46	28.24
ITG-word	35.47	26.55
ITG-mwm (VB)	39.21	<b>29.02</b>
ITG-mwm (EM)	39.15	28.47

Table 1: Translation results on Chinese-English, using the subset of training data (141K sentence pairs) that have length limit 35 on both sides. (No length limit in translation.)

2 points dev set and nearly 1 point of improvement on the test set. We also observe the consistent superiority of VB over EM. The gain is especially large on the test data set, indicating VB is less prone to overfitting.

## 8 Conclusion

We have presented an improved and more efficient method of estimating phrase pairs directly. By both changing the objective function to include a bias toward sparser models and improving the pruning techniques and efficiency, we achieve significant gains on test data with practical speed. In addition, these gains were shown without resorting to external models, such as GIZA++. We have shown that VB is both practical and effective for use in MT models.

However, our best system does not apply VB to a single probability model, as we found an appreciable benefit from bootstrapping each model from simpler models, much as the IBM word alignment models are usually trained in succession. We find that VB alone is not sufficient to counteract the tendency of EM to prefer analyses with smaller trees using fewer rules and longer phrases. Both the tic-tac-toe pruning and the non-compositional constraint address this problem by reducing the space of possible phrase pairs. On top of these hard constraints, the sparse prior of VB helps make the model less prone to overfitting to infrequent phrase pairs, and thus improves the quality of the phrase pairs the model learns.

**Acknowledgments** This work was done while the first author was at Microsoft Research; thanks to Xiaodong He, Mark Johnson, and Kristina Toutanova. The last author was supported by NSF IIS-0546554.

## References

- Matthew Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London.
- Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June.
- Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 17–24, Rochester, New York, April. Association for Computational Linguistics.
- David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, Michigan, USA.
- Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June. Association for Computational Linguistics.
- Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305.
- Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 115–124, Washington, USA, September.
- Kenichi Kurihara and Taisuke Sato. 2006. Variational bayesian grammar induction for natural language. In *International Colloquium on Grammatical Inference*, pages 84–96, Tokyo, Japan.
- Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Robert C. Moore. 2003. Learning translations of named-entity phrases from parallel corpora. In *Proceedings of EACL*, Budapest, Hungary.
- Franz Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March.
- Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, December.
- Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167, Sapporo, Japan.
- Frank Soong and Eng Huang. 1991. A tree-trellis based fast search for finding the n best sentence hypotheses in continuous speech recognition. In *Proceedings of ICASSP 1991*.
- Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING*, pages 836–741, Copenhagen, Denmark.
- Stephan Vogel. 2005. PESA: Phrase pair extraction as sentence splitting. In *MT Summit X*, Phuket, Thailand.
- Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, September.
- Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of ACL*.

# Applying a Grammar-based Language Model to a Simplified Broadcast-News Transcription Task

**Tobias Kaufmann**

Speech Processing Group

ETH Zürich

Zürich, Switzerland

kaufmann@tik.ee.ethz.ch

**Beat Pfister**

Speech Processing Group

ETH Zürich

Zürich, Switzerland

pfister@tik.ee.ethz.ch

## Abstract

We propose a language model based on a precise, linguistically motivated grammar (a hand-crafted Head-driven Phrase Structure Grammar) and a statistical model estimating the probability of a parse tree. The language model is applied by means of an N-best rescoring step, which allows to directly measure the performance gains relative to the baseline system without rescoring. To demonstrate that our approach is feasible and beneficial for non-trivial broad-domain speech recognition tasks, we applied it to a simplified German broadcast-news transcription task. We report a significant reduction in word error rate compared to a state-of-the-art baseline system.

## 1 Introduction

It has repeatedly been pointed out that N-grams model natural language only superficially: an Nth-order Markov chain is a very crude model of the complex dependencies between words in an utterance. More accurate statistical models of natural language have mainly been developed in the field of statistical parsing, e.g. Collins (2003), Charniak (2000) and Ratnaparkhi (1999). Other linguistically inspired language models like Chelba and Jelinek (2000) and Roark (2001) have been applied to continuous speech recognition.

These models have in common that they explicitly or implicitly use a context-free grammar induced from a treebank, with the exception of Chelba and Jelinek (2000). The probability of a rule expansion or parser operation is conditioned on various contextual information and the derivation history. An

important reason for the success of these models is the fact that they are lexicalized: the probability distributions are also conditioned on the actual words occurring in the utterance, and not only on their parts of speech. Most statistical parsers achieve a high robustness with respect to out-of-grammar sentences by allowing for arbitrary derivations and rule expansions. On the other hand, they are not suited to reliably decide on the grammaticality of a given phrase, as they do not accurately model the linguistic constraints inherent in natural language.

We take a completely different position. In the first place, we want our language model to reliably distinguish between grammatical and ungrammatical phrases. To this end, we have developed a precise, linguistically motivated grammar. To distinguish between common and uncommon phrases, we use a statistical model that estimates the probability of a phrase based on the syntactic dependencies established by the parser. We achieve some degree of robustness by letting the grammar accept arbitrary sequences of words and phrases. To keep the grammar restrictive, such sequences are penalized by the statistical model.

Accurate hand-crafted grammars have been applied to speech recognition before, e.g. Kiefer et al. (2000) and van Noord et al. (1999). However, they primarily served as a basis for a speech understanding component and were applied to narrow-domain tasks such as appointment scheduling or public transport information. We are mainly concerned with speech recognition performance on broad-domain recognition tasks.

Beutler et al. (2005) pursued a similar approach.

However, their grammar-based language model did not make use of a probabilistic component, and it was applied to a rather simple recognition task (dictation texts for pupils read and recorded under good acoustic conditions, no out-of-vocabulary words). Besides proposing an improved language model, this paper presents experimental results for a much more difficult and realistic task and compares them to the performance of a state-of-the-art baseline system.

In the following Section, we will first describe our grammar-based language model. Next, we will turn to the linguistic components of the model, namely the grammar, the lexicon and the parser. We will point out some of the challenges arising from the broad-domain speech recognition application and propose ways to deal with them. Finally, we will describe our experiments on broadcast news data and discuss the results.

## 2 Language Model

### 2.1 The General Approach

Speech recognizers choose the word sequence  $\hat{W}$  which maximizes the posterior probability  $P(W|O)$ , where  $O$  is the acoustic observation. This is achieved by optimizing

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(O|W) \cdot P(W)^\lambda \cdot ip^{|W|} \quad (1)$$

The language model weight  $\lambda$  and the word insertion penalty  $ip$  lead to a better performance in practice, but they have no theoretical justification. Our grammar-based language model is incorporated into the above expression as an additional probability  $P_{gram}(W)$ , weighted by a parameter  $\mu$ :

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(O|W) \cdot P(W)^\lambda \cdot P_{gram}(W)^\mu \cdot ip^{|W|} \quad (2)$$

$P_{gram}(W)$  is defined as the probability of the most likely parse tree of a word sequence  $W$ :

$$P_{gram}(W) = \max_{T \in \text{parses}(W)} P(T) \quad (3)$$

To determine  $P_{gram}(W)$  is an expensive operation as it involves parsing. For this reason, we pursue an N-best rescoring approach. We first produce the N best hypotheses according to the criterion in equation (1). From these hypotheses we then choose the final recognition result according to equation (2).

### 2.2 The Probability of a Parse Tree

The parse trees produced by our parser are binary-branching and rather deep. In order to compute the probability of a parse tree, it is transformed to a flat dependency tree similar to the syntax graph representation used in the TIGER treebank Brants et al (2002). An inner node of such a dependency tree represents a constituent or phrase. Typically, it directly connects to a leaf node representing the most important word of the phrase, the *head child*. The other children represent phrases or words directly depending on the head child. To give an example, the immediate children of a sentence node are the finite verb (the head child), the adverbials, the subject and the all other (verbal and non-verbal) complements.

This flat structure has the advantage that the information which is most relevant for the head child is represented within the locality of an inner node. Assuming statistical independence between the internal structures of the inner nodes  $n_i$ , we can factor  $P(T)$  much like it is done for probabilistic context-free grammars:

$$P(T) \approx \prod_{n_i} P(\text{childtags}(n_i) | \text{tag}(n_i)) \quad (4)$$

In the above equation,  $\text{tag}(n_i)$  is simply the label assigned to the tree node  $n_i$ , and  $\text{childtags}(n_i)$  denotes the tags assigned to the child nodes of  $n_i$ .

Our statistical model for German sentences distinguishes between eight different tags. Three tags are used for different types of noun phrases: pronominal NPs, non-pronominal NPs and pronominal genitives. Pronominal genitives were given a dedicated tag because they are much more restricted than ordinary NPs. Another two tags were used to distinguish between clauses with sentence-initial finite verbs (main clauses) and clauses with sentence-final finite verbs (subordinate clauses). Finally, there are specific tags for infinitive verb phrases, adjective phrases and prepositional phrases.

$P$  was modeled by means of a dedicated probability distribution for each conditioning tag. The probability of the internal structure of a sentence was modeled as the trigram probability of the corresponding tag sequence (the sequence of the sentence node's child tags). The probability of an adjective phrase was decomposed into the probability

of the adjective type (participle or non-participle and attributive, adverbial or predicative) and the probability of its length in words given the adjective type. This allows the model to directly penalize long adjective phrases, which are very rare. The model for noun phrases is based on the joint probability of the head type (either noun, adjective or proper name), the presence of a determiner and the presence of pre- and postnominal modifiers. The probabilities of various other events are conditioned on those four variables, namely the number of prepositional phrases, relative clauses and adjectives, as well as the presence of appositions and prenominal or postnominal genitives.

The resulting probability distributions were trained on the German TIGER treebank which consists of about 50000 sentences of newspaper text.

### 2.3 Robustness Issues

A major problem of grammar-based approaches to language modeling is how to deal with out-of-grammar utterances. Obviously, the utterance to be recognized may be ungrammatical, or it could be grammatical but not covered by the given grammar. But even if the utterance is both grammatical and covered by the grammar, the correct word sequence may not be among the  $N$  best hypotheses due to out-of-vocabulary words or bad acoustic conditions. In all these cases, the best hypothesis available is likely to be out-of-grammar, but the language model should nevertheless prefer it to competing hypotheses. To make things worse, it is not unlikely that some of the competing hypotheses are grammatical.

It is therefore important that our language model is robust with respect to out-of-grammar sentences. In particular this means that it should provide a reasonable parse tree for any possible word sequence  $W$ . However, our approach is to use an accurate, linguistically motivated grammar, and it is undesirable to weaken the constraints encoded in the grammar. Instead, we allow the parser to attach any sequence of words or correct phrases to the root node, where each attachment is penalized by the probabilistic model  $P(T)$ . This can be thought of as adding two probabilistic context-free rules:

$$\begin{aligned} S &\longrightarrow S' S && \text{with probability } q \\ S &\longrightarrow S' && \text{with probability } 1-q \end{aligned}$$

In order to guarantee that all possible word sequences are parseable,  $S'$  can produce both saturated phrases and arbitrary words. To include such a productive set of rules into the grammar would lead to serious efficiency problems. For this reason, these rules were actually implemented as a dynamic programming pass: after the parser has identified all correct phrases, the most probable sequence of phrases or words is computed.

### 2.4 Model Parameters

Besides the distributions required to specify  $P(T)$ , our language model has three parameters: the language model weight  $\mu$ , the attachment probability  $q$  and the number of hypotheses  $N$ . The parameters  $\mu$  and  $q$  are considered to be task-dependent. For instance, if the utterances are well-covered by the grammar and the acoustic conditions are good, it can be expected that  $\mu$  is relatively large and that  $q$  is relatively small. The choice of  $N$  is restricted by the available computing power. For our experiments, we chose  $N = 100$ . The influence of  $N$  on the word error rate is discussed in the results section.

## 3 Linguistic Resources

### 3.1 Particularities of the Recognizer Output

The linguistic resources presented in this Section are partly influenced by the form of the recognizer output. In particular, the speech recognizer does not always transcribe numbers, compounds and acronyms as single words. For instance, the word “*einundzwanzig*” (twenty-one) is transcribed as “*ein und zwanzig*”, “*Kriegspläne*” (war plans) as “*Kriegs Pläne*” and “*BMW*” as “*B. M. W.*” These transcription variants are considered to be correct by our evaluation scheme. Therefore, the grammar should accept them as well.

### 3.2 Grammar and Parser

We used the Head-driven Phrase Structure Grammar (HPSG, see Pollard and Sag (1994)) formalism to develop a precise large-coverage grammar for German. HPSG is an unrestricted grammar (Chomsky type 0) which is based on a context-free skeleton and the unification of complex feature structures. There are several variants of HPSG which mainly differ in the formal tools they provide for stating lin-



guistic constraints. Our particular variant requires that constituents (phrases) be continuous, but it provides a mechanism for dealing with discontinuities as present e.g. in the German main clause, see Kaufmann and Pfister (2007). HPSG typically distinguishes between *immediate dominance schemata* (rough equivalents of phrase structure rules, but making no assumptions about constituent order) and *linear precedence rules* (constraints on constituent order). We do not make this distinction but rather let immediate dominance schemata specify constituent order. Further, the formalism allows to express complex linguistic constraints by means of *predicates* or *relational constraints*. At parse time, predicates are backed by program code that can perform arbitrary computations to check or specify feature structures.

We have implemented an efficient Java parser for our variant of the HPSG formalism. The parser supports ambiguity packing, which is a technique for merging constituents with different derivational histories but identical syntactic properties. This is essential for parsing long and ambiguous sentences.

Our grammar incorporates many ideas from existing linguistic work, e.g. Müller (2007), Müller (1999), Crysmann (2005), Crysmann (2003). In addition, we have modeled a few constructions which occur frequently but are often neglected in formal syntactic theories. Among them are prenominal and postnominal genitives, expressions of quantity and expressions of date and time. Further, we have implemented dedicated subgrammars for analyzing written numbers, compounds and acronyms that are written as separate words. To reduce ambiguity, only noun-noun compounds are covered by the grammar. Noun-noun compounds are by far the most productive compound type.

The grammar consists of 17 rules for general linguistic phenomena (e.g. subcategorization, modification and extraction), 12 rules for modeling the German verbal complex and another 13 construction-specific rules (relative clauses, genitive attributes, optional determiners, nominalized adjectives, etc.). The various subgrammars (expressions of date and time, written numbers, noun-noun compounds and acronyms) amount to a total of 43 rules.

The grammar allows the derivation of “intermediate products” which cannot be regarded as complete phrases. We consider complete phrases to be

sentences, subordinate clauses, relative and interrogative clauses, noun phrases, prepositional phrases, adjective phrases and expressions of date and time.

### 3.3 Lexicon

The lexicon was created manually based on a list of more than 5000 words appearing in the N-best lists of our experiment. As the domain of our recognition task is very broad, we attempted to include any possible reading of a given word. Our main source of dictionary information was Duden (1999).

Each word was annotated with precise morphological and syntactic information. For example, the roughly 2700 verbs were annotated with over 7000 valency frames. We distinguish 86 basic valency frames, for most of which the complement types can be further specified.

A major difficulty was the acquisition of *multi-word lexemes*. Slightly deviating from the common notion, we use the following definition: A syntactic unit consisting of two or more words is a multi-word lexeme, if the grammar cannot derive it from its parts. English examples are idioms like “*by and large*” and phrasal verbs such as “*to call sth off*”. Such multi-word lexemes have to be entered into the lexicon, but they cannot directly be identified in the word list. Therefore, they have to be extracted from supplementary resources. For our work, we used a newspaper text corpus of 230M words (Frankfurter Rundschau and Neue Zürcher Zeitung). This corpus included only articles which are dated before the first broadcast news show used in the experiment. In the next few paragraphs we will discuss some types of multiword lexemes and our methods of extracting them.

There is a large and very productive class of German prefix verbs whose prefixes can appear separated from the verb, similar to English phrasal verbs. For example, the prefix of the verb “*untergehen*” (to sink) is separated in “*das Schiff geht unter*” (the ship sinks) and attached in “*weil das Schiff untergeht*” (because the ship sinks). The set of possible valency frames of a prefix verb has to be looked up in a dictionary as it cannot be derived systematically from its parts. Exploiting the fact that prefixes are attached to their verb under certain circumstances, we extracted a list of prefix verbs from the above newspaper text corpus. As the number of prefix verbs is

very large, a candidate prefix verb was included into the lexicon only if there is a recognizer hypothesis in which both parts are present. Note that this procedure does not amount to optimizing on test data: when parsing a hypothesis, the parser chart contains only those multiword lexemes for which all parts are present in the hypothesis.

Other multi-word lexemes are fixed word clusters of various types. For instance, some prepositional phrases appearing in support verb constructions lack an otherwise mandatory determiner, e.g. “*unter Beschuss*” (under fire). Many multi-word lexemes are adverbials, e.g. “*nach wie vor*” (still), “*auf die Dauer*” (in the long run). To extract such word clusters we used suffix arrays proposed in Yamamoto and Church (2001) and the pointwise mutual information measure, see Church and Hanks (1990). Again, it is feasible to consider only those clusters appearing in some recognizer hypothesis. The list of candidate clusters was reduced using different filter heuristics and finally checked manually.

For our task, split compounds are to be considered as multi-word lexemes as well. As our grammar only models noun-noun compounds, other compounds such as “*unionsgeführt*” (led by the union) have to be entered into the lexicon. We applied the decompounding algorithm proposed in Adda-Decker (2003) to our corpus to extract such compounds. The resulting candidate list was again filtered manually.

We observed that many proper nouns (e.g. personal names and geographic names) are identical to some noun, adjective or verb form. For example, about 40% of the nouns in our lexicon share inflected forms with personal names. Proper nouns considerably contribute to ambiguity, as most of them do not require a determiner. Therefore, a proper noun which is a homograph of an open-class word was entered only if it is “relevant” for our task. The “relevant” proper nouns were extracted automatically from our text corpus. We used small databases of unambiguous given names and forms of address to spot personal names in significant bigrams. Relevant geographic names were extracted by considering capitalized words which significantly often follow certain local prepositions.

The final lexicon contains about 2700 verbs (including 1900 verbs with separable prefixes), 3500

nouns, 450 adjectives, 570 closed-class words and 220 multiword lexemes. All lexicon entries amount to a total of 137500 full forms. Noun-noun compounds are not included in these numbers, as they are handled in a morphological analysis component.

## 4 Experiments

### 4.1 Experimental Setup

The experiment was designed to measure how much a given speech recognition system can benefit from our grammar-based language model. To this end, we used a baseline speech recognition system which provided the  $N$  best hypotheses of an utterance along with their respective scores. The grammar-based language model was then applied to the  $N$  best hypotheses as described in Section 2.1, yielding a new best hypothesis. For a given test set we could then compare the word error rate of the baseline system with that of the extended system employing the grammar-based language model.

### 4.2 Data and Preprocessing

Our experiments are based on word lattice output from the LIMSI German broadcast news transcription system (McTait and Adda-Decker, 2003), which employs 4-gram backoff language models. From the experiment reported in McTait and Adda-Decker (2003), we used the first three broadcast news shows<sup>1</sup> which corresponds to a signal length of roughly 50 minutes.

Rather than applying our model to the original broadcast-news transcription task, we used the above data to create an artificial recognition task with manageable complexity. Our primary aim was to design a task which allows us to investigate the properties of our grammar-based approach and to compare its performance with that of a competitive baseline system.

As a first simplification, we assumed perfect sentence segmentation. We manually split the original word lattices at the sentence boundaries and merged them where a sentence crossed a lattice boundary. This resulted in a set of 636 lattices (sentences). Second, we classified the sentences with respect to content type and removed those classes with an excep-

<sup>1</sup>The 8 o'clock broadcasts of the “Tagesschau” from the 14<sup>th</sup> of April, 21<sup>st</sup> of April and 7<sup>th</sup> of Mai 2002.

tionally high baseline word error rate. These classes are interviews (a word error rate of 36.1%), sports reports (28.4%) and press conferences (25.7%). The baseline word error rate of the remaining 447 lattices (sentences) is 11.8%.

From each of these 447 lattices, the 100 best hypotheses were extracted. We next compiled a list containing all words present in the recognizer hypotheses. These words were entered into the lexicon as described in Section 3.3. Finally, all extracted recognizer hypotheses were parsed. Only 25 of the 44000 hypotheses<sup>2</sup> caused an early termination of the parser due to the imposed memory limits. However, the inversion of ambiguity packing (see Section 3.2) turned out to be a bottleneck. As  $P(T)$  does not directly apply to parse trees, all possible readings have to be unpacked. For 24 of the 447 lattices, some of the  $N$  best hypotheses contained phrases with more than 1000 readings. For these lattices the grammar-based language model was simply switched off in the experiment, as no parse trees were produced for efficiency reasons.

To assess the difficulty of our task, we inspected the reference transcriptions, the word lattices and the N-best lists for the 447 selected utterances. We found that for only 59% of the utterances the correct transcription is among the 100-best hypotheses. The first-best hypothesis is completely correct for 34% of the utterances. The out-of-vocabulary rate (estimated from the number of reference transcription words which do not appear in any of the lattices) is 1.7%. The first-best word error rate is 11.79%, and the 100-best oracle word error rate is 4.8%.

We further attempted to judge the grammaticality of the reference transcriptions. We considered only 1% of the sentences to be clearly ungrammatical. 19% of the remaining sentences were found to contain general grammatical constructions which are not handled by our grammar. Some of these constructions (most notably ellipses, which are omnipresent in broadcast-news reports) are notoriously difficult as they would dramatically increase ambiguity when implemented in a grammar. About 45% of the reference sentences were correctly analyzed by the grammar.

---

<sup>2</sup>Some of the word lattices contain less than 100 different hypotheses.

### 4.3 Training and Testing

The parameter  $N$ , the maximum number of hypotheses to be considered, was set to 100 (the effect of choosing different values of  $N$  will be discussed in section 4.4). The remaining parameters  $\mu$  and  $q$  were trained using the leave-one-out cross-validation method: each of the 447 utterances served as the single test item once, whereas the remaining 446 utterances were used for training. As the error landscape is complex and discrete, we could not use gradient-based optimization methods. Instead, we chose  $\mu$  and  $q$  from 500 equidistant points within the intervals  $[0, 20]$  and  $[0, 0.25]$ , respectively. The word error rate was evaluated for each possible pair of parameter values.

The evaluation scheme was taken from McTait and Adda-Decker (2003). It ignores capitalization, and written numbers, compounds and acronyms need not be written as single words.

### 4.4 Results

As shown in Table 1, the grammar-based language model reduced the word error rate by 9.2% relative over the baseline system. This improvement is statistically significant on a level of  $< 0.1\%$  for both the Matched Pairs Sentence-Segment Word Error test (MAPSSWE) and McNemar’s test (Gillick and Cox, 1989). If the parameters are optimized on all 447 sentences (i.e. on the test data), the word error rate is reduced by 10.7% relative.

For comparison, we redefined the probabilistic model as  $P(T) = (1 - q)q^{k-1}$ , where  $k$  is the number of phrases attached to the root node. This reduced model only considers the grammaticality of a phrase, completely ignoring the probability of its internal structure. It achieved a relative word error reduction of 5.9%, which is statistically significant on a level of  $< 0.1\%$  for both tests. The improvement of the full model compared to the reduced model is weakly significant on a level of 2.6% for the MAPSSWE test.

For both models, the optimal value of  $q$  was 0.001 for almost all training runs. The language model weight  $\mu$  of the reduced model was about 60% smaller than the respective value for the full model, which confirms that the full model provides more reliable information.

experiment	word error rate
<b>baseline</b>	<b>11.79%</b>
grammar, no statistics	11.09% (-5.9% rel.)
<b>grammar</b>	<b>10.70%</b> (-9.2% rel.)
grammar, cheating	10.60% (-10.7% rel.)
100-best oracle	4.80%

Table 1: The impact of the grammar-based language model on the word error rate. For comparison, the results for alternative experiments are shown. In the experiment “*grammar, cheating*”, the parameters were optimized on test data.

Figure 1 shows the effect of varying  $N$  (the maximum number of hypotheses) on the word error rate both for leave-one-out training and for optimizing the parameters on test data. The similar shapes of the two curves suggest that the observed variations are partly due to the problem structure. In fact, if  $N$  is increased and new hypotheses with a high value of  $P_{gram}(W)$  appear, the benefit of the grammar-based language model can increase (if the hypotheses are predominantly good with respect to word error rate) or decrease (if they are bad). This horizon effect tends to be reduced with increasing  $N$  (with the exception of  $89 \leq N \leq 93$ ) because hypotheses with high ranks need a much higher  $P_{gram}(W)$  in order to compensate for their lower value of  $P(O|W) \cdot P(W)^\lambda$ . For small  $N$ , the parameter estimation is more severely affected by the rather accidental horizon effects and therefore is prone to overfitting.

## 5 Conclusions and Outlook

We have presented a language model based on a precise, linguistically motivated grammar, and we have successfully applied it to a difficult broad-domain task.

It is a well-known fact that natural language is highly ambiguous: a correct and seemingly unambiguous sentence may have an enormous number of readings. A related – and for our approach even more relevant – phenomenon is that many weird-looking and seemingly incorrect word sequences are in fact grammatical. This obviously reduces the benefit of pure grammaticality information. A solution is to use additional information to assess how “natural” a reading of a word sequence is. We have done a

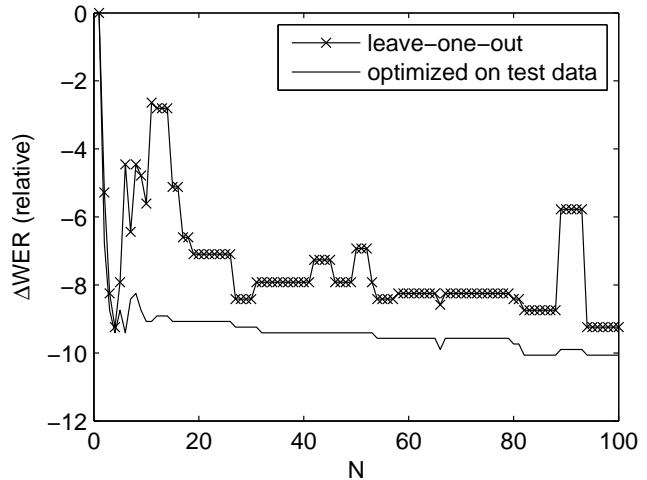


Figure 1: The word error rate as a function of the maximum number of best hypotheses  $N$ .

first step in this direction by estimating the probability of a parse tree. However, our model only looks at the structure of a parse tree and does not take the actual words into account. As N-grams and statistical parsers demonstrate, word information can be very valuable. It would therefore be interesting to investigate ways of introducing word information into our grammar-based model.

## Acknowledgements

This work was supported by the Swiss National Science Foundation. We cordially thank Jean-Luc Gauvain of LIMSI for providing us with word lattices from their German broadcast news transcription system.

## References

- M. Adda-Decker. 2003. A corpus-based decomposing algorithm for German lexical modeling in LVCSR. In *Proceedings of Eurospeech*, pages 257–260, Geneva, Switzerland.
- R. Beutler, T. Kaufmann, and B. Pfister. 2005. Integrating a non-probabilistic grammar into large vocabulary continuous speech recognition. In *Proceedings of the IEEE ASRU 2005 Workshop*, pages 104–109, San Juan (Puerto Rico).
- S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, Bulgaria.
- E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the NAACL*, pages 132–139, San Francisco, USA.
- C. Chelba and F. Jelinek. 2000. Structured language modeling. *Computer Speech & Language*, 14(4):283–332.
- K. W. Church and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.
- M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.
- B. Crysmann. 2003. On the efficient implementation of German verb placement in HPSG. In *Proceedings of RANLP*.
- B. Crysmann. 2005. Relative clause extraposition in German: An efficient and portable implementation. *Research on Language and Computation*, 3(1):61–82.
- Duden. 1999. – *Das große Wörterbuch der deutschen Sprache in zehn Bänden*. Dudenverlag, dritte Auflage.
- L. Gillick and S. Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of the ICASSP*, pages 532–535.
- T. Kaufmann and B. Pfister. 2007. Applying licenser rules to a grammar with continuous constituents. In Stefan Müller, editor, *The Proceedings of the 14th International Conference on Head-Driven Phrase Structure Grammar*, pages 150–162, Stanford, USA. CSLI Publications.
- B. Kiefer, H.-U. Krieger, and M.-J. Nederhof. 2000. Efficient and robust parsing of word hypotheses graphs. In Wolfgang Wahlster, editor, *Verbmobil. Foundations of Speech-to-Speech Translation*, pages 280–295. Springer, Berlin, Germany, artificial intelligence edition.
- K. McTait and M. Adda-Decker. 2003. The 300k LIMSI German broadcast news transcription system. In *Proceedings of Eurospeech*, Geneva, Switzerland.
- S. Müller. 1999. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Number 394 in *Linguistische Arbeiten*. Max Niemeyer Verlag, Tübingen.
- S. Müller. 2007. *Head-Driven Phrase Structure Grammar: Eine Einführung*. Stauffenburg Einführungen, Nr. 17. Stauffenburg Verlag, Tübingen.
- G. Van Noord, G. Bouma, R. Koeling, and M.-J. Nederhof. 1999. Robust grammatical analysis for spoken dialogue systems. *Natural Language Engineering*, 5(1):45–93.
- C. J. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.
- A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.
- B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.
- M. Yamamoto and K. 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.

# Automatic Editing in a Back-End Speech-to-Text System

Maximilian Bisani Paul Vozila Olivier Divay Jeff Adams

Nuance Communications

One Wayside Road

Burlington, MA 01803, U.S.A.

{maximilian.bisani,paul.vozila,olivier.divay,jeff.adams}@nuance.com

## Abstract

Written documents created through dictation differ significantly from a true verbatim transcript of the recorded speech. This poses an obstacle in automatic dictation systems as speech recognition output needs to undergo a fair amount of editing in order to turn it into a document that complies with the customary standards. We present an approach that attempts to perform this edit from recognized words to final document automatically by learning the appropriate transformations from example documents. This addresses a number of problems in an integrated way, which have so far been studied independently, in particular automatic punctuation, text segmentation, error correction and disfluency repair. We study two different learning methods, one based on rule induction and one based on a probabilistic sequence model. Quantitative evaluation shows that the probabilistic method performs more accurately.

## 1 Introduction

Large vocabulary speech recognition today achieves a level of accuracy that makes it useful in the production of written documents. Especially in the medical and legal domains large volumes of text are traditionally produced by means of dictation. Here document creation is typically a “back-end” process. The author dictates all necessary information into a telephone handset or a portable recording device and is not concerned with the actual production of the document any further. A transcriptionist will then

listen to the recorded dictation and produce a well-formed document using a word processor. The goal of introducing speech recognition in this process is to create a draft document automatically, so that the transcriptionist only has to verify the accuracy of the document and to fix occasional recognition errors. We observe that users try to spend as little time as possible dictating. They usually focus only on the content and rely on the transcriptionist to compose a readable, syntactically correct, stylistically acceptable and formally compliant document. For this reason there is a considerable discrepancy between the final document and what the speaker has said literally. In particular in medical reports we see differences of the following kinds:

- Punctuation marks are typically not verbalized.
- No instructions on the formatting of the report are dictated. Section headings are not identified as such.
- Frequently section headings are only implied. (“vitals are” → “PHYSICAL EXAMINATION: VITAL SIGNS:”)
- Enumerated lists. Typically speakers use phrases like “number one . . . next number . . .”, which need to be turned into “1. . . 2. . .”
- The dictation usually begins with a preamble (e.g. “This is doctor Xyz ...”) which does not appear in the report. Similarly there are typical phrases at the end of the dictation which should not be transcribed (e.g. “End of dictation. Thank you.”)

- There are specific standards regarding the use of medical terminology. Transcriptionists frequently expand dictated abbreviations (e.g. “CVA” → “cerebrovascular accident”) or otherwise use equivalent terms (e.g. “nonicteric sclerae” → “no scleral icterus”).
- The dictation typically has a more narrative style (e.g. “She has no allergies.”, “I examined him”). In contrast, the report is normally more impersonal and structured (e.g. “ALLERGIES: None.”, “he was examined”).
- For the sake of brevity, speakers frequently omit function words. (“patient” → “the patient”, “denies fever pain” → “he denies any fever or pain”)
- As the dictation is spontaneous, disfluencies are quite frequent, in particular false starts, corrections and repetitions. (e.g. “22-year-old female, sorry, male 22-year-old male” → “22-year-old male”)
- Instruction to the transcriptionist and so-called normal reports, pre-defined text templates invoked by a short phrase like “This is a normal chest x-ray.”
- In addition to the above, speech recognition output has the usual share of recognition errors some of which may occur systematically.

These phenomena pose a problem that goes beyond the speech recognition task which has traditionally focused on correctly identifying speech utterances. Even with a perfectly accurate verbatim transcript of the user’s utterances, the transcriptionist would need to perform a significant amount of editing to obtain a document conforming to the customary standards. We need to look for what the user wants rather than what he says.

Natural language processing research has addressed a number of these issues as individual problems: automatic punctuation (Liu et al., 2005), text segmentation (Beeferman et al., 1999; Matusov et al., 2003) disfluency repair (Heeman et al., 1996) and error correction (Ringger and Allen, 1996; Strzalkowski and Brandow, 1997; Peters and Drexel,

2004). The method we present in the following attempts to address all this by a unified transformation model. The goal is simply stated as transforming the recognition output into a text document. We will first describe the general framework of learning transformations from example documents. In the following two sections we will discuss a rule-induction-based and a probabilistic transformation method respectively. Finally we present experimental results in the context of medical transcription and conclude with an assessment of both methods.

## 2 Text transformation

In dictation and transcription management systems corresponding pairs of recognition output and edited and corrected documents are readily available. The idea of transformation modeling, outlined in figure 1, is to learn to emulate the transcriptionist. To this end we first process archived dictations with the speech recognizer to create approximate verbatim transcriptions. For each document this yields the spoken or *source* word sequence  $S = s_1 \dots s_M$ , which is supposed to be a word-by-word transcription of the user’s utterances, but which may actually contain recognition errors. The corresponding final reports are cleaned (removal of page headers etc.), tagged (identification of section headings and enumerated lists) and tokenized, yielding the text or *target* token sequence  $T = t_1 \dots t_N$  for each document. Generally, the token sequence corresponds to the spoken form. (E.g. “25mg” is tokenized as “twenty five milligrams”.) Tokens can be ordinary words or special symbols representing line breaks, section headings, etc. Specifically, we represent each section heading by a single indivisible token, even if the section name consists of multiple words. Enumerations are represented by special tokens, too. Different techniques can be applied to learn and execute the actual transformation from  $S$  to  $T$ . Two options are discussed in the following.

With the transformation model at hand, a draft for a new document is created in three steps. First the speech recognizer processes the audio recording and produces the source word sequence  $S$ . Next, the transformation step converts  $S$  into the target sequence  $T$ . Finally the transformation output  $T$  is formatted into a text document. Formatting is the

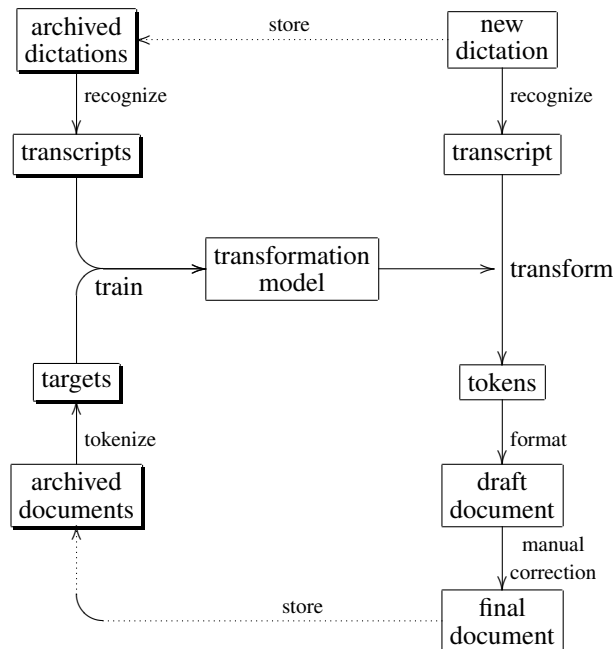


Figure 1: Illustration of how text transformation is integrated into a speech-to-text system.

inverse of tokenization and includes conversion of number words to digits, rendition of paragraphs and section headings, etc.

Before we turn to concrete transformation techniques, we can make two general statements about this problem. Firstly, in the absence of observations to the contrary, it is reasonable to leave words unchanged. So, a priori the mapping should be the identity. Secondly, the transformation is mostly monotonous. Out-of-order sections do occur but are the exception rather than the rule.

### 3 Transformation based learning

Following Strzalkowski and Brandow (1997) and Peters and Drexel (2004) we have implemented a *transformation-based learning* (TBL) algorithm (Brill, 1995). This method iteratively improves the match (as measured by token error rate) of a collection of corresponding source and target token sequences by positing and applying a sequence of *substitution rules*. In each iteration the source and target tokens are aligned using a minimum edit distance criterion. We refer to maximal contiguous subsequences of non-matching tokens as *error re-*

*gions*. These consist of paired sequences of source and target tokens, where either sequence may be empty. Each error region serves as a candidate substitution rule. Additionally we consider refinements of these rules with varying amounts of contiguous context tokens on either side. Deviating from Peters and Drexel (2004), in the special case of an empty target sequence, i.e. a deletion rule, we consider deleting all (non-empty) contiguous subsequences of the source sequence as well. For each candidate rule we accumulate two counts: the number of exactly matching error regions and the number of false alarms, i.e. when its left-hand-side matches a sequence of already correct tokens. Rules are ranked by the difference in these counts scaled by the number of errors corrected by a single rule application, which is the length of the corresponding error region. This is an approximation to the total number of errors corrected by a rule, ignoring rule interactions and non-local changes in the minimum edit distance alignment. A subset of the top-ranked non-overlapping rules satisfying frequency and minimum impact constraints are selected and the source sequences are updated by applying the selected rules. Again deviating from Peters and Drexel (2004), we consider two rules as overlapping if the left-hand-side of one is a contiguous subsequence of the other. This procedure is iterated until no additional rules can be selected. The initial rule set is populated by a small sequence of hand-crafted rules (e.g. “impression colon” → “IMPRESSION:”). A user-independent baseline rule set is generated by applying the algorithm to data from a collection of users. We construct speaker-dependent models by initializing the algorithm with the speaker-independent rule set and applying it to data from the given user.

### 4 Probabilistic model

The canonical approach to text transformation following statistical decision theory is to maximize the text document posterior probability given the spoken document.

$$T^* = \operatorname{argmax}_T p(T|S) \quad (1)$$

Obviously, the global model  $p(T|S)$  must be constructed from smaller scale observations on the cor-



responsiveness between source and target words. We use a 1-to-n alignment scheme. This means each source word is assigned to a sequence of zero, one or more target words. We denote the target words assigned to source word  $s_i$  as  $\tau_i$ . Each replacement  $\tau_i$  is a possibly empty sequence of target words. A source word together with its replacement sequence will be called a *segment*. We constrain the set of possible transformations by selecting a relatively small set of allowable replacements  $A(s)$  to each source word. This means we require  $\tau_i \in A(s_i)$ . We use the usual  $m$ -gram approximation to model the joint probability of a transformation:

$$p(S, T) = \prod_{i=1}^M p(s_i, \tau_i | s_{i-m+1}, \tau_{i-m+1}, \dots, s_{i-1}, \tau_{i-1}) \quad (2)$$

The work of Ringger and Allen (1996) is similar in spirit to this method, but uses a factored source-channel model. Note that the decision rule (1) is over whole documents. Therefore we process complete documents at a time without prior segmentation into sentences.

To estimate this model we first align all training documents. That is, for each document, the target word sequence is segmented into  $M$  segments  $T = \tau_1 \cup \dots \cup \tau_M$ . The criterion for this alignment is to maximize the likelihood of a segment unigram model. The alignment is performed by an expectation maximization algorithm. Subsequent to the alignment step,  $m$ -gram probabilities are estimated by standard language modeling techniques. We create speaker-specific models by linearly interpolating an  $m$ -gram model based on data from the user with a speaker-independent background  $m$ -gram model trained on data pooled from a collection of users.

To select the allowable replacements for each source word we count how often each particular target sequence is aligned to it in the training data. A source target pair is selected if it occurs twice or more times. Source words that were not observed in training are immutable, i.e. the word itself is its only allowable replacement  $A(s) = \{s\}$ . As an example suppose “patient” was deleted 10 times, left unchanged 105 times, replaced by “the patient” 113 times and once replaced by “she”. The word patient would then have three allowables:  $A(\text{patient}) = \{(), (\text{patient}), (\text{the, patient})\}$ .

The decision rule (1) minimizes the document error rate. A more appropriate loss function is the number of source words that are replaced incorrectly. Therefore we use the following *minimum word risk* (MWR) decision strategy, which minimizes source word loss.

$$T^* = (\operatorname{argmax}_{\tau_1 \in A(s_1)} p(\tau_1 | S)) \cup \dots \cup (\operatorname{argmax}_{\tau_M \in A(s_M)} p(\tau_M | S)) \quad (3)$$

This means for each source sequence position we choose the replacement that has the highest posterior probability  $p(\tau_i | S)$  given the entire source sequence. To compute the posterior probabilities, first a graph is created representing alternatives “around” the most probable transform using beam search. Then the forward-backward algorithm is applied to compute edge posterior probabilities. Finally edge posterior probabilities for each source position are accumulated.

## 5 Experimental evaluation

The methods presented were evaluated on a set of real-life medical reports dictated by 51 doctors. For each doctor we use 30 reports as a test set. Transformation models are trained on a disjoint set of reports that predated the evaluation reports. The typical document length is between one hundred and one thousand words. All dictations were recorded via telephone. The speech recognizer works with acoustic models that are specifically adapted for each user, not using the test data, of course. It is hard to quote the verbatim word error rate of the recognizer, because this would require a careful and time-consuming manual transcription of the test set. The recognition output is auto-punctuated by a method similar in spirit to the one proposed by Liu et al. (2005) before being passed to the transformation model. This was done because we considered the auto-punctuation output as the status quo ante which transformation modeling was to be compared to. Neither of both transformation methods actually relies on having auto-punctuated input. The auto-punctuation step only inserts periods and commas and the document is not explicitly segmented into sentences. (The transformation step always applies to entire documents and the interpretation of a period as a sentence boundary is left to the human

Table 1: Experimental evaluation of different text transformation techniques with different amounts of user-specific data. Precision, recall, deletion, insertion and error rate values are given in percent and represent the average of 51 users, where the results for each user are the ratios of sums over 30 reports.

method	user docs	sections		punctuation		all tokens		
		precision	recall	precision	recall	deletions	insertions	errors
none (only auto-punct)		0.00	0.00	66.68	71.21	11.32	27.48	45.32
TBL	SI	69.18	44.43	73.90	67.22	11.41	17.73	34.99
3-gram	SI	65.19	44.41	73.79	62.26	18.15	12.27	36.09
TBL	25	75.38	53.39	75.59	69.11	10.97	15.97	32.62
3-gram	25	80.90	59.37	78.88	69.81	11.50	12.09	28.87
TBL	50	76.67	56.18	76.11	69.81	10.81	15.53	31.92
3-gram	50	81.10	62.69	79.39	70.94	11.31	11.46	27.76
TBL	100	77.92	58.03	76.41	70.52	10.67	15.19	31.29
3-gram	100	81.69	64.36	79.35	71.38	11.48	10.82	27.12
3-gram without MWR	100	81.39	64.23	79.01	71.52	11.55	10.92	27.29

reader of the document.) For each doctor a background transformation model was constructed using 100 reports from each of the *other* users. This is referred to as the speaker-independent (SI) model. In the case of the probabilistic model, all models were 3-gram models. User-specific models were created by augmenting the SI model with 25, 50 or 100 reports. One report from the test set is shown as an example in the appendix.

## 5.1 Evaluation metric

The output of the text transformation is aligned with the corresponding tokenized report using a minimum edit cost criterion. Alignments between section headings and non-section headings are not permitted. Likewise no alignment of punctuation and non-punctuation tokens is allowed. Using the alignment we compute precision and recall for sections headings and punctuation marks as well as the overall token error rate. It should be noted that the so derived error rate is not comparable to word error rates usually reported in speech recognition research. All missing or erroneous section headings, punctuation marks and line breaks are counted as errors. As pointed out in the introduction the reference texts do not represent a literal transcript of the dictation. Furthermore the data were not cleaned manually. There are, for example, instances of letter heads or page numbers that were not correctly removed when the text was extracted from the word processor’s file for-

mat. The example report shown in the appendix features some of the typical differences between the produced draft and the final report that may or may not be judged as errors. (For example, the date of the report was not given in the dictation, the section names “laboratory data” and “laboratory evaluation” are presumably equivalent and whether “stable” is preceded by a hyphen or a period in the last section might not be important.) Nevertheless, the numbers reported do permit a quantitative comparison between different methods.

## 5.2 Results

Results are stated in table 1. In the baseline setup no transformation is applied to the auto-punctuated recognition output. Since many parts of the source data do not need to be altered, this constitutes the reference point for assessing the benefit of transformation modeling. For obvious reasons precision and recall of section headings are zero. A high rate of insertion errors is observed which can largely be attributed to preambles. Both transformation methods reduce the discrepancy between the draft document and the final corrected document significantly. With 100 training documents per user the mean token error rate is reduced by up to 40% relative by the probabilistic model. When user specific data is used, the probabilistic approach performs consistently better than TBL on all accounts. In particular it always has much lower insertion rates reflecting its supe-

rior ability to remove utterances that are not typically part of the report. On the other hand the probabilistic model suffers from a slightly higher deletion rate due to being overzealous in this regard. In speaker independent mode, however, the deletion rate is excessively high and leads to inferior overall performance. Interestingly the precision of the automatic punctuation is increased by the transformation step, without compromising on recall, at least when enough user specific training data is available. The minimum word risk criterion (3) yields slightly better results than the simpler document risk criterion (1).

## 6 Conclusions

Automatic text transformation brings speech recognition output much closer to the end result desired by the user of a back-end dictation system. It automatically punctuates, sections and rephrases the document and thereby greatly enhances transcriptionist productivity. The holistic approach followed here is simpler and more comprehensive than a cascade of more specialized methods. Whether or not the holistic approach is also more accurate is not an easy question to answer. Clearly the outcome would depend on the specifics of the specialized methods one would compare to, as well as the complexity of the integrated transformation model one applies. The simple models studied in this work admittedly have little provisions for targeting specific transformation problems. For example the typical length of a section is not taken into account. However, this is not a limitation of the general approach. We have observed that a simple probabilistic sequence model performs consistently better than the transformation-based learning approach. Even though neither of both methods is novel, we deem this an important finding since none of the previous publications we know of in this domain allow this conclusion. While the present experiments have used a separate auto-punctuation step, future work will aim to eliminate it by integrating the punctuation features into the transformation step. In the future we plan to integrate additional knowledge sources into our statistical method in order to more specifically address each of the various phenomena encountered in spontaneous dictation.

## References

- Beeferman, Doug, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177 – 210.
- Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543 – 565.
- Heeman, Peter A., Kyung-ho Loken-Kim, and James F. Allen. 1996. Combining the detection and correction of speech repairs. In *Proc. Int. Conf. Spoken Language Processing (ICSLP)*, pages 362 – 365. Philadelphia, PA, USA.
- Liu, Yang, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *Proc. Annual Meeting of the ACL*, pages 451 – 458. Ann Arbor, MI, USA.
- Matusov, Evgeny, Jochen Peters, Carsten Meyer, and Hermann Ney. 2003. Topic segmentation using markov models on section level. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 471 – 476. IEEE, St. Thomas, U.S. Virgin Islands.
- Peters, Jochen and Christina Drexel. 2004. Transformation-based error correction for speech-to-text systems. In *Proc. Int. Conf. Spoken Language Processing (ICSLP)*, pages 1449 – 1452. Jeju Island, Korea.
- Ringger, Eric K. and James F. Allen. 1996. A fertility channel model for post-correction of continuous speech recognition. In *Proc. Int. Conf. Spoken Language Processing (ICSLP)*, pages 897 – 900. Philadelphia, PA, USA.
- Strzalkowski, Tomek and Ronald Brandow. 1997. A natural language correction model for continuous speech recognition. In *Proc. 5th Workshop on Very Large Corpora (WVLC-5)*., pages 168 – 177. Beijing-Hong Kong.

## Appendix A. Example of a medical report

<p>Recognition output. Vertical space was added to facilitate visual comparison.</p>	<p>Automatically generated draft (speech recognition output after transformation and formatting)</p>	<p>Final report produced by a human transcriptionist without reference to the automatic draft.</p>
<p>doctors name dictating a progress note on first name last name patient without complaints has been ambulating without problems no chest pain chest pressure still has some shortness of breath but overall has improved significantly</p> <p>vital signs are stable she is afebrile lungs show decreased breath sounds at the bases with bilateral rales and rhonchi heart is regular rate and rhythm two over six crescendo decrescendo murmur at the right sternal border abdomen soft nontender nondistended extremities show one plus pedal edema bilaterally neurological exam is nonfocal</p> <p>white count of five point seven H. and H. eleven point six and thirty five point five platelet count of one fifty five sodium one thirty seven potassium three point nine chloride one hundred carbon dioxide thirty nine calcium eight point seven glucose ninety one BUN and creatinine thirty seven and one point one</p> <p>impression number one COPD exacerbation continue breathing treatments number two asthma exacerbation continue oral prednisone number three bronchitis continue Levaquin number four hypertension stable improved number six gastroesophageal reflux disease stable number seven congestive heart failure stable</p> <p>new paragraph patient is in stable condition and will be discharged to name nursing home and will be monitored closely on an outpatient basis progress note</p>	<p>Progress note</p> <p>SUBJECTIVE: The patient is without complaints. Has been ambulating without problems. No chest pain, chest pressure, still has some shortness of breath, but overall has improved significantly.</p> <p>PHYSICAL EXAMINATION:</p> <p>VITAL SIGNS: Stable. She is afebrile.</p> <p>LUNGS: Show decreased breath sounds at the bases with bilateral rales and rhonchi.</p> <p>HEART: Regular rate and rhythm 2/6 crescendo decrescendo murmur at the right sternal border.</p> <p>ABDOMEN: Soft, nontender, nondistended.</p> <p>EXTREMITIES: Show 1+ pedal edema bilaterally.</p> <p>NEUROLOGICAL: Nonfocal.</p> <p>LABORATORY DATA: White count of 5.7, hemoglobin and hematocrit 11.6 and 35.5, platelet count of 155, sodium 137, potassium 3.9, chloride 100, CO2 39, calcium 8.7, glucose 91, BUN and creatinine 37 and 1.1.</p> <p>IMPRESSION:</p> <ol style="list-style-type: none"> <li>1. Chronic obstructive pulmonary disease exacerbation. Continue breathing treatments.</li> <li>2. Asthma exacerbation. Continue oral prednisone.</li> <li>3. Bronchitis. Continue Levaquin.</li> <li>4. Hypertension. Stable.</li> <li>5. Uncontrolled diabetes mellitus. Improved.</li> <li>6. Gastroesophageal reflux disease, stable.</li> <li>7. Congestive heart failure. Stable.</li> </ol> <p>PLAN: The patient is in stable condition and will be discharged to name nursing home and will be monitored closely on an outpatient basis.</p>	<p>Progress Note</p> <p>DATE: July 26, 2005.</p> <p>HISTORY OF PRESENT ILLNESS: The patient has no complaints. She is ambulating without problems. No chest pain or chest pressure. She still has some shortness of breath, but overall has improved significantly.</p> <p>PHYSICAL EXAMINATION:</p> <p>VITAL SIGNS: Stable. She's afebrile.</p> <p>LUNGS: Decreased breath sounds at the bases with bilateral rales and rhonchi.</p> <p>HEART: Regular rate and rhythm. 2/6 crescendo, decrescendo murmur at the right sternal border.</p> <p>ABDOMEN: Soft, nontender and nondistended.</p> <p>EXTREMITIES: 1+ pedal edema bilaterally.</p> <p>NEUROLOGICAL EXAMINATION: Nonfocal.</p> <p>LABORATORY EVALUATION: White count 5.7, H&amp;H 11.6 and 35.5, platelet count of 155, sodium 137, potassium 3.9, chloride 100, co2 39, calcium 8.7, glucose 91, BUN and creatinine 37 and 1.1.</p> <p>IMPRESSION:</p> <ol style="list-style-type: none"> <li>1. Chronic obstructive pulmonary disease exacerbation. Continue breathing treatments.</li> <li>2. Asthma exacerbation. Continue oral prednisone.</li> <li>3. Bronchitis. Continue Levaquin.</li> <li>4. Hypertension-stable.</li> <li>5. Uncontrolled diabetes mellitus-improved.</li> <li>6. Gastroesophageal reflux disease-stable.</li> <li>7. Congestive heart failure-stable.</li> </ol> <p>The patient is in stable condition and will be discharged to name Nursing Home, and will be monitored on an outpatient basis.</p>

# Grounded Language Modeling for Automatic Speech Recognition of Sports Video

**Michael Fleischman**

Massachusetts Institute of Technology  
Media Laboratory  
mbf@mit.edu

**Deb Roy**

Massachusetts Institute of Technology  
Media Laboratory  
dkroy@media.mit.edu

## Abstract

Grounded language models represent the relationship between words and the non-linguistic context in which they are said. This paper describes how they are learned from large corpora of unlabeled video, and are applied to the task of automatic speech recognition of sports video. Results show that grounded language models improve perplexity and word error rate over text based language models, and further, support video information retrieval better than human generated speech transcriptions.

## 1 Introduction

Recognizing speech in broadcast video is a necessary precursor to many multimodal applications such as video search and summarization (Snoek and Worring, 2005;). Although performance is often reasonable in controlled environments (such as studio news rooms), automatic speech recognition (ASR) systems have significant difficulty in noisier settings (such as those found in live sports broadcasts) (Wactlar et al., 1996). While many researches have examined how to compensate for such noise using acoustic techniques, few have attempted to leverage information in the visual stream to improve speech recognition performance (for an exception see Murkherjee and Roy, 2003).

In many types of video, however, visual context can provide valuable clues as to what has been said. For example, in video of Major League Baseball games, the likelihood of the phrase “home run” increases dramatically when a home run has actually been hit. This paper describes a method for incorporating such visual information in an ASR system for sports video. The method is based on the use of *grounded language models* to repre-

sent the relationship between words and the non-linguistic context to which they refer (Fleischman and Roy, 2007).

Grounded language models are based on research from cognitive science on grounded models of meaning. (for a review see Roy, 2005, and Roy and Reiter, 2005). In such models, the meaning of a word is defined by its relationship to representations of the language users’ environment. Thus, for a robot operating in a laboratory setting, words for colors and shapes may be grounded in the outputs of its computer vision system (Roy & Pentland, 2002); while for a simulated agent operating in a virtual world, words for actions and events may be mapped to representations of the agent’s plans or goals (Fleischman & Roy, 2005).

This paper extends previous work on grounded models of meaning by learning a grounded language model from naturalistic data collected from broadcast video of Major League Baseball games. A large corpus of unlabeled sports videos is collected and paired with closed captioning transcriptions of the announcers’ speech.<sup>1</sup> This corpus is used to train the grounded language model, which like traditional language models encode the prior probability of words for an ASR system. Unlike traditional language models, however, grounded language models represent the probability of a word conditioned not only on the previous word(s), but also on features of the non-linguistic context in which the word was uttered.

Our approach to learning grounded language models operates in two phases. In the first phase, events that occur in the video are represented using hierarchical temporal pattern automatically mined

---

<sup>1</sup> Closed captioning refers to human transcriptions of speech embedded in the video stream primarily for the hearing impaired. Closed captioning is reasonably accurate (although not perfect) and available on some, but not all, video broadcasts.

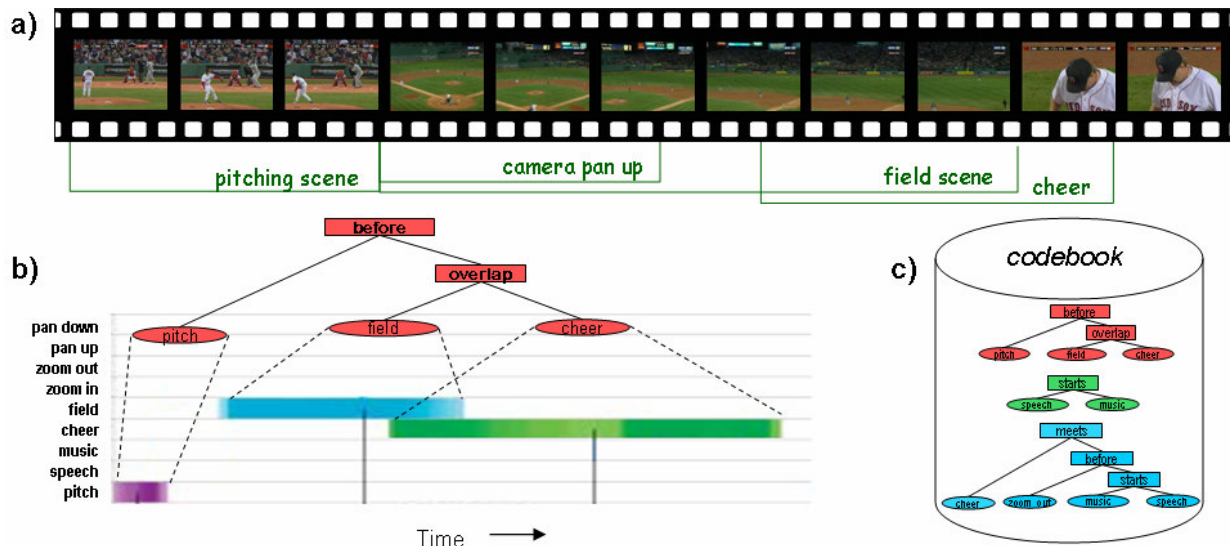


Figure 1. Representing events in video. a) Events are represented by first abstracting the raw video into visual context, camera motion, and audio context features. b) Temporal data mining is then used to discover hierarchical temporal patterns in the parallel streams of features. c) Temporal patterns found significant in each iteration are stored in a codebook that is used to represent high level events in video.

from low level features. In the second phase, a conditional probability distribution is estimated that describes the probability that a word was uttered given such event representations. In the following sections we describe these two aspects of our approach and evaluate the performance of our grounded language model on a speech recognition task using video highlights from Major League Baseball games. Results indicate improved performance using three metrics: perplexity, word error rate, and precision on an information retrieval task.

## 2 Representing Events in Sports Video

Recent work in video surveillance has demonstrated the benefit of representing complex events as temporal relations between lower level sub-events (Hongen et al., 2004). Thus, to represent events in the sports domain, we would ideally first represent the basic sub events that occur in sports video (e.g., hitting, throwing, catching, running, etc.) and then build up complex events (such as *home run*) as a set of temporal relations between these basic events. Unfortunately, due to the limitations of computer vision techniques, reliably identifying such basic events in video is not feasible. However, sports video does have characteristics that can be exploited to effectively represent complex events.

Like much broadcast video, sports video is highly produced, exploiting many different camera angles and a human director who selects which camera is most appropriate given what is happening on the field. The styles that different directors employ are extremely consistent within a sport and make up a “language of film” which the machine can take advantage of in order to represent the events taking place in the video.

Thus, even though it is not easy to automatically identify a player hitting a ball in video, it is easy to detect features that correlate with hitting, e.g., when a scene focusing on the pitching mound immediately jumps to one zooming in on the field (see Figure 1). Although these correlations are not perfect, experiments have shown that baseball events can be classified using such features (Fleischman et al., 2007).

We exploit the language of film to represent events in sports video in two phases. First, low level features that correlate with basic events in sports are extracted from the video stream. Then, temporal data mining is used to find patterns within this low level event stream.

### 2.1 Feature Extraction

We extract three types of features: visual context features, camera motion features, and audio context features.

## Visual Context Features

Visual context features encode general properties of the visual scene in a video segment. Supervised classifiers are trained to identify these features, which are relatively simple to classify in comparison to high level events (like home runs) that require more training data and achieve lower accuracy. The first step in classifying visual context features is to segment the video into shots (or scenes) based on changes in the visual scene due to editing (e.g. jumping from a close up to a wide shot of the field). Shot detection and segmentation is a well studied problem; in this work we use the method of Tardini et al. (2005).

After the video is segmented into shots, individual frames (called key frames) are selected and represented as a vector of low level features that describe the key frame's color distribution, entropy, etc. (see Fleischman and Roy, 2007 for the full list of low level features used). The WEKA machine learning package is used to train a boosted decision tree to classify these frames into one of three categories: *pitching-scene*, *field-scene*, *other* (Witten and Frank, 2005). Those shots whose key frames are classified as *field-scenes* are then sub-categorized (using boosted decision trees) into one of the following categories: *infield*, *outfield*, *wall*, *base*, *running*, and *misc*. Performance of these classification tasks is approximately 96% and 90% accuracy respectively.

## Camera Motion Features

In addition to visual context features, we also examine the camera motion that occurs within a video. Unlike visual context features, which provide information about the global situation that is being observed, camera motion features represent more precise information about the actions occurring in a video. The intuition here is that the camera is a stand in for a viewer's focus of attention. As actions occur in a video, the camera moves to follow it; this camera motion thus mirrors the actions themselves, providing informative features for event representation.

Like shot boundary detection, detecting the motion of the camera in a video (i.e., the amount it pans left to right, tilts up and down, and zooms in and out) is a well-studied problem. We use the system of Bouthemy et al. (1999) which computes the camera motion using the parameters of a two-

dimensional affine model to fit every pair of sequential frames in a video. A 15 state 1<sup>st</sup> order Hidden Markov Model, implemented with the Graphical Modeling Toolkit,<sup>2</sup> then converts the output of the Bouthemy system into a stream of clustered characteristic camera motions (e.g. state 12 clusters together motions of zooming in fast while panning slightly left).

## Audio Context

The audio stream of a video can also provide useful information for representing non-linguistic context. We use boosted decision trees to classify audio into segments of *speech*, *excited\_speech*, *cheering*, and *music*. Classification operates on a sequence of overlapping 30 ms frames extracted from the audio stream. For each frame, a feature vector is computed using, MFCCs (often used in speaker identification and speech detection tasks), as well as energy, the number of zero crossings, spectral entropy, and relative power between different frequency bands. The classifier is applied to each frame, producing a sequence of class labels. These labels are then smoothed using a dynamic programming cost minimization algorithm (similar to those used in Hidden Markov Models). Performance of this system achieves between 78% and 94% accuracy.

## 2.2 Temporal Pattern Mining

Given a set of low level features that correlate with the basic events in sports, we can now focus on building up representations of complex events. Unlike previous work (Hongen et al., 2005) in which representations of the temporal relations between low level events are built up by hand, we employ temporal data mining techniques to automatically discover such relations from a large corpus of unannotated video.

As described above, ideal basic events (such as hitting and catching) cannot be identified easily in sports video. By finding temporal patterns between audio, visual and camera motion features, however, we can produce representations that are highly correlated with sports events. Importantly, such temporal patterns are not strictly sequential, but rather, are composed of features that can occur

---

<sup>2</sup> <http://ssli.ee.washington.edu/~bilmes/gmtk/>

in complex and varied temporal relations to each other.

To find such patterns automatically, we follow previous work in video content classification in which temporal data mining techniques are used to discover event patterns within streams of lower level features. The algorithm we use is fully unsupervised and proceeds by examining the relations that occur between features in multiple streams within a moving time window. Any two features that occur within this window must be in one of seven temporal relations with each other (e.g. *before*, *during*, *etc.*) (Allen, 1984). The algorithm keeps track of how often each of these relations is observed, and after the entire video corpus is analyzed, uses chi-square analyses to determine which relations are significant. The algorithm iterates through the data, and relations between individual features that are found significant in one iteration (e.g. [OVERLAP, *field-scene*, *cheer*]), are themselves treated as individual features in the next. This allows the system to build up higher-order nested relations in each iteration (e.g. [BEFORE, [OVERLAP, *field-scene*, *cheer*], *field scene*]).

The temporal patterns found significant in this way make up a codebook which can then be used as a basis for representing a video. The term codebook is often used in image analysis to describe a set of features (stored in the codebook) that are used to encode raw data (images or video). Such codebooks are used to represent raw video using features that are more easily processed by the computer.

Our framework follows a similar approach in which raw video is encoded (using a codebook of temporal patterns) as follows. First, the raw video is abstracted into the visual context, camera motion, and audio context feature streams (as described in Section 2.1). These feature streams are then scanned, looking for any temporal patterns (and nested sub-patterns) that match those found in the codebook. For each pattern, the duration for which it occurs in the feature streams is treated as the value of an element in the vector representation for that video.

Thus, a video is represented as an  $n$  length vector, where  $n$  is the total number of temporal patterns in the codebook. The value of each element of this vector is the duration for which the pattern associated with that element was observed in the video. So, if a pattern was not observed in a video

at all, it would have a value of 0, while if it was observed for the entire length of the video, it would have a value equal to the number of frames present in that video.

Given this method for representing the non-linguistic context of a video, we can now examine how to model the relationship between such context and the words used to describe it.

### 3 Linguistic Mapping

Modeling the relationship between words and non-linguistic context assumes that the speech uttered in a video refers consistently (although not exclusively) to the events being represented by the temporal pattern features. We model this relationship, much like traditional language models, using conditional probability distributions. Unlike traditional language models, however, our grounded language models condition the probability of a word not only on the word(s) uttered before it, but also on the temporal pattern features that describe the non-linguistic context in which it was uttered. We estimate these conditional distributions using a framework similar that used for training acoustic models in ASR and translation models in Machine Translation (MT).

We generate a training corpus of utterances paired with representations of the non-linguistic context in which they were uttered. The first step in generating this corpus is to generate the low level features described in Section 2.1 for each video in our training set. We then segment each video into a set of independent events based on the visual context features we have extracted. We follow previous work in sports video processing (Gong et al., 2004) and define an event in a baseball video as any sequence of shots starting with a *pitching-scene* and continuing for four subsequent shots. This definition follows from the fact that the vast majority of events in baseball start with a pitch and do not last longer than four shots. For each of these events in our corpus, a temporal pattern feature vector is generated as described in section 2.2. These events are then paired with all the words from the closed captioning transcription that occur during each event (plus or minus 10 seconds). Because these transcriptions are not necessarily time synched with the audio, we use the method described in Hauptmann and Witbrock



(1998) to align the closed captioning to the announcers’ speech.

Previous work has examined applying models often used in MT to the paired corpus described above (Fleischman and Roy, 2006). Recent work in automatic image annotation (Barnard et al., 2003; Blei and Jordan, 2003) and natural language processing (Steyvers et al., 2004), however, have demonstrated the advantages of using hierarchical Bayesian models for related tasks. In this work we follow closely the Author-Topic (AT) model (Steyvers et al., 2004) which is a generalization of Latent Dirichlet Allocation (LDA) (Blei et al., 2005).<sup>3</sup>

LDA is a technique that was developed to model the distribution of topics discussed in a large corpus of documents. The model assumes that every document is made up of a mixture of topics, and that each word in a document is generated from a probability distribution associated with one of those topics. The AT model generalizes LDA, saying that the mixture of topics is not dependent on the document itself, but rather on the authors who wrote it. According to this model, for each word (or phrase) in a document, an author is chosen uniformly from the set of the authors of the document. Then, a topic is chosen from a distribution of topics associated with that particular author. Finally, the word is generated from the distribution associated with that chosen topic. We can express the probability of the words in a document ( $W$ ) given its authors ( $A$ ) as:

$$p(W | A) = \prod_{m \in W} \frac{1}{A_d} \sum_{x \in A} \sum_{z \in T} p(m | z) p(z | x) \quad (1)$$

where  $T$  is the set of latent topics that are induced given a large set of training data.

We use the AT model to estimate our grounded language model by making an analogy between documents and events in video. In our framework, the words in a document correspond to the words in the closed captioning transcript associated with an event. The authors of a document correspond to the temporal patterns representing the non-

linguistic context of that event. We modify the AT model slightly, such that, instead of selecting from

<sup>3</sup> In the discussion that follows, we describe a method for estimating unigram grounded language models. Estimating bigram and trigram models can be done by processing on word pairs or triples, and performing normalization on the resulting conditional distributions.

a uniform distribution (as is done with authors of documents), we select patterns from a multinomial distribution based upon the duration of the pattern. The intuition here is that patterns that occur for a longer duration are more salient and thus, should be given greater weight in the generative process. We can now rewrite (1) to give the probability of words during an event ( $W$ ) given the vector of observed temporal patterns ( $P$ ) as:

$$p(W | P) = \prod_{m \in W} \sum_{x \in P} \sum_{z \in T} p(m | z) p(z | x) p(x) \quad (2)$$

In the experiments described below we follow Steyver et al., (2004) and train our AT model using Gibbs sampling, a Markov Chain Monte Carlo technique for obtaining parameter estimates. We run the sampler on a single chain for 200 iterations. We set the number of topics to 15, and normalize the pattern durations first by individual pattern across all events, and then for all patterns within an event. The resulting parameter estimates are smoothed using a simple add  $N$  smoothing technique, where  $N=1$  for the word by topic counts and  $N=.01$  for the pattern by topic counts.

## 4 Evaluation

In order to evaluate our grounded language modeling approach, a parallel data set of 99 Major League Baseball games with corresponding closed captioning transcripts was recorded from live television. These games represent data totaling approximately 275 hours and 20,000 distinct events from 25 teams in 23 stadiums, broadcast on five different television stations. From this set, six games were held out for testing (15 hours, 1200 events, nine teams, four stations). From this test set, baseball highlights (i.e., events which terminate with the player either *out* or *safe*) were hand annotated for use in evaluation, and manually transcribed in order to get clean text transcriptions for gold standard comparisons. Of the 1200 events in the test set, 237 were highlights with a total word count of 12,626 (vocabulary of 1800 words).

The remaining 93 unlabeled games are used to train unigram, bigram, and trigram grounded language models. Only unigrams, bigrams, and trigrams that are not proper names, appear greater than three times, and are not composed only of stop words were used. These grounded language models are then combined in a backoff strategy

with traditional unigram, bigram, and trigram language models generated from a combination of the closed captioning transcripts of all training games and data from the switchboard corpus (see below). This backoff is necessary to account for the words not included in the grounded language model itself (i.e. stop words, proper names, low frequency words). The traditional text-only language models (which are also used below as baseline comparisons) are generated with the SRI language modeling toolkit (Stolcke, 2002) using Chen and Goodman's modified Kneser-Ney discounting and interpolation (Chen and Goodman, 1998). The backoff strategy we employ here is very simple: if the ngram appears in the GLM then it is used, otherwise the traditional LM is used. In future work we will examine more complex backoff strategies (Hsu, in review).

We evaluate our grounded language modeling approach using 3 metrics: perplexity, word error rate, and precision on an information retrieval task.

#### 4.1 Perplexity

Perplexity is an information theoretic measure of how well a model predicts a held out test set. We use perplexity to compare our grounded language model to two baseline language models: a language model generated from the switchboard corpus, a commonly used corpus of spontaneous speech in the telephony domain (3.65M words; 27k vocab); and a language model that interpolates (with equal weight given to both) between the switchboard model and a language model trained only on the baseball-domain closed captioning (1.65M words; 17k vocab). The results of calculating perplexity on the test set highlights for these three models is presented in Table 1 (lower is better).

Not surprisingly, the switchboard language model performs far worse than both the interpolated text baseline and the grounded language model. This is due to the large discrepancy between both the style and vocabulary of language about sports compared to the domain of telephony sampled by the switchboard corpus. Of more interest is the decrease in perplexity seen when using the grounded language model compared to the interpolated model. Note that these two language models are generated using the same speech transcriptions, i.e. the closed captioning from the training games and the switchboard corpus. However,

whereas the baseline model remains the same for each of the 237 test highlights, the grounded language model generates different word distributions for each highlight depending on the event features extracted from the highlight video.

	Switchboard	Interpolated (Switch+CC)	Grounded
ppl	1404	145.27	83.88

Table 1. Perplexity measures for three different language models on a held out test set of baseball highlights (12,626 words). We compare the grounded language model to two text based language models: one trained on the switchboard corpus alone; and interpolated with one trained on closed captioning transcriptions of baseball video.

#### 4.2 Word Accuracy and Error Rate

Word error rate (WER) is a normalized measure of the number of word insertions, substitutions, and deletions required to transform the output transcription of an ASR system to a human generated gold standard transcription of the same utterance. Word accuracy is simply the number of words in the gold standard that they system correctly recognized. Unlike perplexity which only evaluates the performance of language models, examining word accuracy and error rate requires running an entire ASR system, i.e. both the language and acoustic models.

We use the Sphinx system to train baseball specific acoustic models using parallel acoustic/text data automatically mined from our training set. Following Jang and Hauptman (1999), we use an off the shelf acoustic model (the hub4 model) to generate an extremely noisy speech transcript of each game in our training set, and use dynamic programming to align these noisy outputs to the closed captioning stream for those same games. Given these two transcriptions, we then generate a paired acoustic/text corpus by sampling the audio at the time codes where the ASR transcription matches the closed captioning transcription.

For example, if the ASR output contains the term sequence "... and farther *home run for David* forty says..." and the closed captioning contains the sequence "...another *home run for David* Ortiz....," the matched phrase "*home run for David*" is assumed a correct transcription for the audio at the time codes given by the ASR system. Only looking at sequences of three words or more,

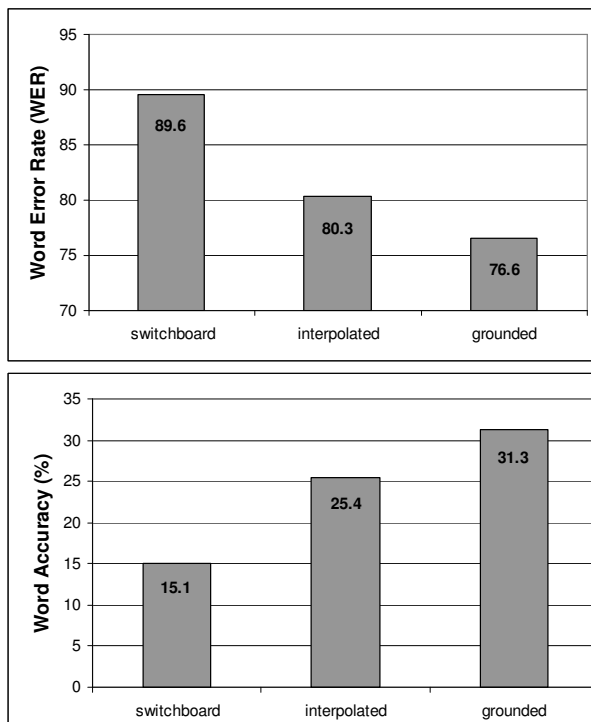


Figure 3. Word accuracy and error rates for ASR systems using a grounded language model, a text based language model trained on the switchboard corpus, and the switchboard model interpolated with a text based model trained on baseball closed captions.

we extract approximately 18 hours of clean paired data from our 275 hour training corpus. A continuous acoustic model with 8 gaussians and 6000 ties states is trained on this data using the Sphinx speech recognizer.<sup>4</sup>

Figure 3 shows the WERs and accuracy for three ASR systems run using the Sphinx decoder with the acoustic model described above and either the grounded language model or the two baseline models described in section 4.1. Note that performance for all of these systems is very poor due to limited acoustic data and the large amount of background crowd noise present in sports video (and particularly in sports highlights). Even with this noise, however, results indicate that the word accuracy and error rates when using the grounded language model is significantly better than both the switchboard model (absolute WER reduction of 13%; absolute accuracy increase of 15.2%) and the switchboard interpolated with the baseball specific text based language model (absolute WER reduction of 3.7%; absolute accuracy increase of 5.9%).

<sup>4</sup> <http://cmusphinx.sourceforge.net/html/cmusphinx.php>

Drawing conclusions about the usefulness of grounded language models using word accuracy or error rate alone is difficult. As it is defined, these measures penalizes a system that mistakes “a” for “uh” as much as one that mistakes “run” for “rum.” When using ASR to support multimedia applications (such as search), though, such substitutions are not of equal importance. Further, while visual information may be useful for distinguishing the latter error, it is unlikely to assist with the former. Thus, in the next section we examine an extrinsic evaluation in which grounded language models are judged not directly on their effect on word accuracy or error rate, but based on their ability to support video information retrieval.

### 4.3 Precision of Information Retrieval

One of the most commonly used applications of ASR for video is to support information retrieval (IR). Such video IR systems often use speech transcriptions to index segments of video in much the same way that words are used to index text documents (Wactlar et al., 1996). For example, in the domain of baseball, if a video IR system were issued the query “home run,” it would typically return a set of video clips by searching its database for events in which someone uttered the phrase “home run.” Because such systems rely on ASR output to search video, the performance of a video IR system gives an indirect evaluation of the ASR’s quality. Further, unlike the case with word accuracy or error rate, such evaluations highlight a systems ability to recognize the more relevant content words without being distracted by the more common stop words.

Our metric for evaluation is the precision with which baseball highlights are returned in a video IR system. We examine three systems: one that uses ASR with the grounded language model, a baseline system that uses ASR with the text only interpolated language model, and finally a system that uses human produced closed caption transcriptions to index events.

For each system, all 1200 events from the test set (not just the highlights) are indexed. Queries are generated artificially using a method similar to Berger and Lafferty (1999) and used in Fleischman and Roy (2007). First, each highlight is labeled with the event’s type (e.g. *fly ball*), the event’s location (e.g. *left field*) and the event’s result (e.g. *double play*): 13 labels total. Log likelihood ratios

are then used to find the phrases (unigram, trigram, and bigram) most indicative of each label (e.g. “fly ball” for category *fly ball*). For each label, the three most indicative phrases are issued as queries to the system, which ranks its results using the language modeling approach of Ponte and Croft (1998). Precision is measured on how many of the top five returned events are of the correct category.

Figure 4 shows the precision of the video IR systems based on ASR with the grounded language model, ASR with the text-only interpolated language model, and closed captioning transcriptions. As with our previous evaluations, the IR results show that the system using ASR with the grounded language model performed better than the one using ASR with the text-only language model (5.1% absolute improvement). More notably, though, Figure 4 shows that the system using the grounded language model performed better than the system using the hand generated closed captioning transcriptions (4.6% absolute improvement). Although this is somewhat counterintuitive given that hand transcriptions are typically considered gold standards, these results follow from a limitation of using text-based methods to index video.

Unlike the case with text documents, the occurrence of a query term in a video is often not enough to assume the video’s relevance to that query. For example, when searching through video of baseball games, returning all clips in which the phrase “home run” occurs, results primarily in video of events where a home run does not actually occur. This follows from the fact that in sports, as in life, people often talk not about what is currently happening, but rather, they talk about what did, might, or will happen in the future.

By taking into account non-linguistic context during speech recognition, the grounded language model system indirectly circumvents some of these false positive results. This follows from the fact that an effect of using the grounded language model is that when an announcer utters a phrase (e.g., “fly ball”), the system is more likely to recognize that phrase correctly if the event it refers to is actually occurring (e.g. if someone actually hit a fly ball). Because the grounded language model system is biased to recognize phrases that describe what is currently happening, it returns fewer false positives and gets higher precision.

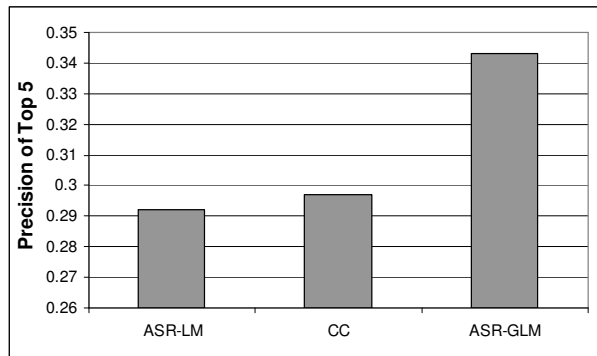


Figure 4. Precision of top five results of a video IR system based on speech transcriptions. Three different transcriptions are compared: ASR-LM uses ASR with a text-only interpolated language model (trained on baseball closed captioning and the switchboard corpus); ASR-GLM uses ASR with a grounded language model; CC uses human generated closed captioning transcriptions (i.e., no ASR).

## 5 Conclusions

We have described a method for improving speech recognition in video. The method uses grounded language modeling, an extension of traditional language modeling in which the probability of a word is conditioned not only on the previous word(s) but also on the non-linguistic context in which the word is uttered. Context is represented using hierarchical temporal patterns of low level features which are mined automatically from a large unlabeled video corpus. Hierarchical Bayesian models are then used to map these representations to words. Initial results show grounded language models improve performance on measures of perplexity, word accuracy and error rate, and precision on an information retrieval task.

In future work, we will examine the ability of grounded language models to improve performance for other natural language tasks that exploit text based language models, such as Machine Translation. Also, we are examining extending this approach to other sports domains such as American football. In theory, however, our approach is applicable to any domain in which there is discussion of the here-and-now (e.g., cooking shows, etc.). In future work, we will examine the strengths and limitations of grounded language modeling in these domains.

## References

- Allen, J.F. (1984). A General Model of Action and Time. *Artificial Intelligence*, 23(2).
- Barnard, K, Duygulu, P, de Freitas, N, Forsyth, D, Blei, D, and Jordan, M. (2003), Matching Words and Pictures, *Journal of Machine Learning Research*, Vol 3.
- Berger, A. and Lafferty, J. (1999). Information Retrieval as Statistical Translation. In *Proceedings of SIGIR-99*.
- Blei, D. and Jordan, M. (2003). Modeling annotated data. *Proceedings of the 26<sup>th</sup> International Conference on Research and Development in Information Retrieval*, ACM Press, 127–134.
- Blei, D. Ng, A., and Jordan, M (2003). “Latent Dirichlet allocation.” *Journal of Machine Learning Research* 3:993–1022.
- Bouthemy, P., Gelgon, M., Ganansia, F. (1999). A unified approach to shot change detection and camera motion characterization. *IEEE Trans. on Circuits and Systems for Video Technology*, 9(7).
- Chen, S. F. and Goodman, J., (1998). An Empirical Study of Smoothing Techniques for Language Modeling, Tech. Report TR-10-98, Computer Science Group, Harvard U., Cambridge, MA.
- Fleischman M, Roy, D. (2007). Situated Models of Meaning for Sports Video Retrieval. *HLT/NAACL*. Rochester, NY.
- Fleischman, M. and Roy, D. (2007). Unsupervised Content-Based Indexing of Sports Video Retrieval. *9<sup>th</sup> ACM Workshop on Multimedia Information Retrieval (MIR)*. Augsburg, Germany.
- Fleischman, M. B. and Roy, D. (2005) Why Verbs are Harder to Learn than Nouns: Initial Insights from a Computational Model of Intention Recognition in Situated Word Learning. *27th Annual Meeting of the Cognitive Science Society*, Stresa, Italy.
- Fleischman, M., DeCamp, P. Roy, D. (2006). Mining Temporal Patterns of Movement for Video Content Classification. *ACM Workshop on Multimedia Information Retrieval*.
- Fleischman, M., Roy, B., and Roy, D. (2007). Temporal Feature Induction for Sports Highlight Classification. In *Proceedings of ACM Multimedia*. Augsburg, Germany.
- Gong, Y., Han, M., Hua, W., Xu, W. (2004). Maximum entropy model-based baseball highlight detection and classification. *Computer Vision and Image Understanding*, 96(2).
- Hauptmann, A., Witbrock, M., (1998) Story Segmentation and Detection of Commercials in Broadcast News Video, *Advances in Digital Libraries*.
- Hongen, S., Nevatia, R. Bremond, F. (2004). Video-based event recognition: activity representation and probabilistic recognition methods. *Computer Vision and Image Understanding*, 96(2).
- Hsu, Bo-June (Paul). (in review). Generalized Linear Interpolation of Language Models.
- Jang, P., Hauptmann, A. (1999). Learning to Recognize Speech by Watching Television. *IEEE Intelligent Systems Magazine*, 14(5), pp. 51-58.
- Mukherjee, N. and Roy, D.. (2003). A Visual Context-Aware Multimodal System for Spoken Language Processing. *Proc. Eurospeech*, 4 pages.
- Ponte, J.M., and Croft, W.B. (1998). A Language Modeling Approach to Information Retrieval. In *Proc. of SIGIR’98*.
- Roy, D. (2005). . Grounding Words in Perception and Action: Insights from Computational Models. *TICS*.
- Roy, D. and Pentland, A. (2002). Learning Words from Sights and Sounds: A Computational Model. *Cognitive Science*, 26(1).
- Roy, D. and Reiter, E. (2005). . Connecting Language to the World. *Artificial Intelligence*, 167(1-2), 1-12.
- Snoek, C.G.M. and Worring, M.. (2005). Multimodal video indexing: A review of the state-of-the-art. *Multimedia Tools and Applications*, 25(1):5-35.
- Steyvers, M., Smyth, P., Rosen-Zvi, M., & Griffiths, T. (2004). Probabilistic Author-Topic Models for Information Discovery. *The Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Seattle, Washington.
- Stolcke, A., (2002). SRILM - An Extensible Language Modeling Toolkit, in *Proc. Intl. Conf. Spoken Language Processing*, Denver, Colorado.
- Tardini, G. Grana C., Marchi, R., Cucchiara, R., (2005). Shot Detection and Motion Analysis for Automatic MPEG-7 Annotation of Sports Videos. In *13th International Conference on Image Analysis and Processing*.
- Wactlar, H., Witbrock, M., Hauptmann, A., (1996 ). Informedia: News-on-Demand Experiments in Speech Recognition. *ARPA Speech Recognition Workshop*, Arden House, Harriman, NY.
- Witten, I. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. 2<sup>nd</sup> Edition, Morgan Kaufmann. San Francisco, CA.

# Lexicalized phonotactic word segmentation

Margaret M. Fleck

Department of Computer Science

University of Illinois

Urbana, IL 61801, USA

mfleck@cs.uiuc.edu

## Abstract

This paper presents a new unsupervised algorithm (WordEnds) for inferring word boundaries from transcribed adult conversations. Phone ngrams before and after observed pauses are used to bootstrap a simple discriminative model of boundary marking. This fast algorithm delivers high performance even on morphologically complex words in English and Arabic, and promising results on accurate phonetic transcriptions with extensive pronunciation variation. Expanding training data beyond the traditional miniature datasets pushes performance numbers well above those previously reported. This suggests that WordEnds is a viable model of child language acquisition and might be useful in speech understanding.

## 1 Introduction

Words are essential to most models of language and speech understanding. Word boundaries define the places at which speakers can fluently pause, and limit the application of most phonological rules. Words are a key constituent in structural analyses: the output of morphological rules and the constituents in syntactic parsing. Most speech recognizers are word-based. And, words are entrenched in the writing systems of many languages.

Therefore, it is generally accepted that children learning their first language must learn how to segment speech into a sequence of words. Similar, but more limited, learning occurs when adults hear speech containing unfamiliar words. These words must be accurately delimited, so that they can be

added to the lexicon and nearby familiar words recognized correctly. Current speech recognizers typically misinterpret such speech.

This paper will consider algorithms which segment phonetically transcribed speech into words. For example, Figure 1 shows a transcribed phrase from the Buckeye corpus (Pitt et al., 2005; Pitt et al., 2007) and the automatically segmented output. Like almost all previous researchers, I use human-transcribed input to work around the limitations of current speech recognizers.

In most available datasets, words are transcribed using standard dictionary pronunciations (henceforth “dictionary transcriptions”). These transcriptions are approximately phonemic and, more importantly, assign a constant form to each word. I will also use one dataset with accurate phonetic transcriptions, including natural variation in the pronunciation of words. Handling this variation is an important step towards eventually using phone lattices or features produced by real speech recognizers.

This paper will focus on segmentation of speech between adults. This is the primary input for speech recognizers. Moreover, understanding such speech is the end goal of child language acquisition. Models tested only on simplified child-directed speech are incomplete without an algorithm for upgrading the understander to handle normal adult speech.

## 2 The task in more detail

This paper uses a simple model of the segmentation task, which matches prior work and the available datasets. Possible enhancements to the model are discussed at the end.

```

"all the kids in there      # are people that have kids # or that are having kids"
IN  REAL: ohlThikidsinner   # ahrpiyp@lThA?HAvkids    # ohrThADurHAViynqkids
    DICT: ahlThiykidzinTher # ahrpiyp@lThAtHAvkidz    # owrThAtahrHAVinqkidz
OUT REAL: ohl Thi kids inner # ahr piyp@l ThA? HAV kids # ohr ThADur HAViynq kids
    DICT: ahl Thiy kidz in Ther # ahr piyp@l ThAt HAV kidz # owr ThAt ahr HAVinq kidz

```

Figure 1: Part of Buckeye corpus dialog 2101a, in accurate phonetic transcription (REAL) and dictionary pronunciations (DICT). Both use modified arpabet, with # marking pauses. Notice the two distinct pronunciations of “that” in the accurate transcription. Automatically inserted word boundaries are shown at bottom.

## 2.1 The input data

This paper considers only languages with an established tradition of words, e.g. not Chinese. I assume that the authors of each corpus have given us reasonable phonetic transcriptions and word boundaries. The datasets are informal conversations in which debatable word segmentations are rare.

The transcribed data is represented as a sequence of phones, with neither prosodic/stress information nor feature representations for the phones. These phone sequences are presented to segmentation algorithms as strings of ASCII characters. Large phonesets may be represented using capital letters and punctuation or, more readably, using multi-character phone symbols. Well-designed (e.g. easily decodable) multi-character codes do not affect the algorithms or evaluation metrics in this paper. Testing often also uses orthographic datasets.

Finally, the transcriptions are divided into “phrases” at pauses in the speech signal (silences, breaths, etc). These pause phrases are **not** necessarily syntactic or prosodic constituents. Disfluencies in conversational speech create pauses where you might not expect them, e.g. immediately following the definite article (Clark and Wasow, 1998; Fox Tree and Clark, 1997). Therefore, I have chosen corpora in which pauses have been marked carefully.

## 2.2 Affixes and syllables

A theory of word segmentation must explain how affixes differ from free-standing function words. For example, we must explain why English speakers consider “the” to be a word, but “-ing” to be an affix, although neither occurs by itself in fluent prepared English. We must also explain why the Arabic determiner “Al-” is not a word, though its syntactic and semantic role seems similar to English “the”.

Viewed another way, we must show how to esti-

mate the average word length. Conversational English has short words (about 3 phones), because most grammatical morphemes are free-standing. Languages with many affixes have longer words, e.g. my Arabic data averages 5.6 phones per word.

Pauses are vital for deciding what is an affix. Attempts to segment transcriptions without pauses, e.g. (Christiansen et al., 1998), have worked poorly. Claims that humans can extract words without pauses seem to be based on psychological experiments such as (Saffran, 2001; Jusczyk and Aslin, 1995) which conflate words and morphemes. Even then, explicit boundaries seem to improve performance (Seidl and Johnson, 2006).

Another significant part of this task is finding syllable boundaries. For English, many phone strings have multiple possible syllabifications. Because words average only 1.26 syllables, segmenting pre-syllabified input has a very high baseline: 100% precision and 80% recall of boundary positions.

## 2.3 Algorithm testing

Unsupervised algorithms are presented with the transcription, divided only at phrase boundaries. Their task is to infer the phrase-internal word boundaries. The primary worry in testing is that development may have biased the algorithm towards a particular language, speaking style, and/or corpus size. Addressing this requires showing that different corpora can be handled with a common set of parameter settings. Therefore a test/training split within one corpus serves little purpose and is not standard.

Supervised algorithms are given training data with all word boundaries marked, and must infer word boundaries in a separate test set. Simple supervised algorithms perform extremely well (Cairns et al., 1997; Teahan et al., 2000), but don’t address our main goal: **learning** how to segment.

Notice that phrase boundaries are not randomly

selected word boundaries. Syntactic and communicative constraints make pauses more likely at certain positions than others. Therefore, the “supervised” algorithms for this task train on a representative set of word boundaries whereas “unsupervised” algorithms train on a biased set of word boundaries. Moreover, supplying **all** the word boundaries for even a small amount of data effectively tells the supervised algorithms the average word length, a parameter which is otherwise not easy to estimate.

Standard evaluation metrics include the precision, recall and F-score <sup>1</sup> of the phrase-internal boundaries (BP, BR, BF), of the extracted word tokens (WP, WR, WF), and of the resulting lexicon of word types (LP, LR, LF). Outputs don’t look good until BF is at least 90%.

### 3 Previous work

Learning to segment words is an old problem, with extensive prior work surveyed in (Batchelder, 2002; Brent and Cartwright, 1996; Cairns et al., 1997; Goldwater, 2006; Hockema, 2006; Rytting, 2007). There are two major approaches. *Phonotactic* methods model which phone sequences are likely within words and which occur primarily across or adjacent to word boundaries. *Language modelling* methods build word ngram models, like those used in speech recognition. Statistical criteria define the “best” model fitting the input data. In both cases, details are complex and variable.

#### 3.1 Phonotactic Methods

Supervised phonotactic methods date back at least to (Lamel and Zue, 1984), see also (Harrington et al., 1989). Statistics of phone trigrams provide sufficient information to segment adult conversational speech (dictionary transcriptions with simulated phonology) with about 90% precision and 93% recall (Cairns et al., 1997), see also (Hockema, 2006). Teahan et al.’s compression-based model (2000) achieves BF over 99% on orthographic English. Segmentation by adults is sensitive to phonotactic constraints (McQueen, 1998; Weber, 2000).

To build unsupervised algorithms, Brent and Cartwright suggested (1996) inferring phonotactic constraints from phone sequences observed at

<sup>1</sup> $F = \frac{2PR}{P+R}$  where  $P$  is the precision and  $R$  is the recall.

phrase boundaries. However, experimental results are poor. Early results using neural nets by Cairns et al. (1997) and Christiansen et al (1998) are discouraging. Rytting (2007) seems to have the best result: 61.0% boundary recall with 60.3% precision <sup>2</sup> on 26K words of modern Greek data, average word length 4.4 phones. This algorithm used mutual information plus phrase-final 2-phone sequences. He obtained similar results (Rytting, 2004) using phrase-final 3-phone sequences.

Word segmentation experiments by Christiansen and Allen (1997) and Harrington et al. (1989). simulated the effects of pronunciation variation and/or recognizer error. Rytting (2007) uses actual speech recognizer output. These experiments broke useful new ground, but poor algorithm performance (BF  $\leq$  50% even on dictionary transcriptions) makes it hard to draw conclusions from their results.

#### 3.2 Language modelling methods

So far, language modelling methods have been more effective. Brent (1999) and Venkataraman (2001) present incremental splitting algorithms with BF about 82% <sup>3</sup> on the Bernstein-Ratner (BR87) corpus of infant-directed English with disfluencies and interjections removed (Bernstein Ratner, 1987; Brent, 1999). Batchelder (2002) achieved almost identical results using a clustering algorithm. The most recent algorithm (Goldwater, 2006) achieves a BF of 85.8% using a Dirichlet Process bigram model, estimated using a Gibbs sampling algorithm.<sup>4</sup>

Language modelling methods incorporate a bias towards re-using hypothesized words. This suggests they should systematically segment morphologically complex words, so as to exploit the structure they share with other words. Goldwater, the only author to address this issue explicitly, reports that her algorithm breaks off common affixes (e.g. “ing”, “s”). Batchelder reports a noticeable drop in performance on Japanese data, which might relate to its more complex words (average 4.1 phones).

<sup>2</sup>These numbers have been adjusted so as not to include boundaries between phrases.

<sup>3</sup>Numbers are from Goldwater’s (2006) replication.

<sup>4</sup>Goldwater numbers are from the December 2007 version of her code, with its suggested parameter values:  $\alpha_0 = 3000$ ,  $\alpha_1 = 300$ ,  $p\# = 0.2$ .



## 4 The new approach

Previous algorithms have modelled either whole words or very short (e.g. 2-3) phone sequences. The new approach proposed in this paper, “lexicalized phonotactics,” models extended sequences of phones at the starts and ends of word sequences. This allows a new algorithm, called WordEnds, to successfully mark word boundaries with a simple local classifier.

### 4.1 The idea

This method models sequences of phones that start or end at a word boundary. When words are long, such a sequence may cover only part of the word e.g. a group of suffixes or a suffix plus the end of the stem. A sequence may also include parts of multiple short words, capturing some simple bits of syntax.

These longer sequences capture not only purely phonotactic constraints, but also information about the inventory of lexical items. This improves handling of complex, messy inputs. (Cf. Ando and Lee’s (2000) kanji segmenter.)

On the other hand, modelling only partial words helps the segmenter handle long, infrequent words. Long words are typically created by productive morphology and, thus, often start and end just like other words. Only 32% of words in Switchboard occur both before and after pauses, but many of the other 68% have similar-looking beginnings or endings.

Given an inter-character position in a phrase, its *right and left contexts* are the character sequences to its right and left. By convention, phrases input to WordEnds are padded with a single blank at each end. So the middle position of the phrase “afunjoke” has right context “joke□” and left context “□afun.” Since this is a word boundary, the right context looks like the start of a real word sequence, and the left context looks like the end of one. This is not true for the immediately previous position, which has right context “njoke□” and left context “□afu.”

Boundaries will be marked where the right and left contexts look like what we have observed at the starts and ends of phrases.

### 4.2 Statistical model

To formalize this, consider a fixed inter-character position in a phrase. It may be a word boundary ( $b$ )

or not ( $-b$ ). Let  $r$  and  $l$  be its right and left contexts. The input data will (see Section 4.3) give us  $P(b|r)$  and  $P(b|l)$ . Deciding whether to mark a boundary at this position requires estimating  $P(b|r, l)$ .

To express  $P(b|r, l)$  in terms of  $P(b|l)$  and  $P(b|r)$ , I will assume that  $r$  and  $l$  are conditionally independent given  $b$ . This corresponds roughly to a unigram language model. Let  $P(b)$  be the probability of a boundary at a random inter-character position. I will assume that the average word length, and therefore  $P(b)$ , is not absurdly small or large.

$P(b|r, l)$  is  $\frac{P(r, l|b)P(b)}{P(r, l)}$ . Conditional independence implies that this is  $\frac{P(r|b)P(l|b)P(b)}{P(r, l)}$ , which is  $\frac{P(r)P(b|r)P(l)P(b|l)}{P(b)P(r, l)}$ . This is  $\frac{P(b|r)P(b|l)}{QP(b)}$  where  $Q = \frac{P(r, l)}{P(r)P(l)}$ .  $Q$  is typically not 1, because a right and left context often co-occur simply because they both tend to occur at boundaries.

To estimate  $Q$ , write  $P(r, l)$  as  $P(r, l, b) + P(r, l, -b)$ . Then  $P(r, l, b)$  is  $\frac{P(r)P(b|r)P(l)P(b|l)}{P(b)}$ . If we assume that  $r$  and  $l$  are also conditionally independent given  $-b$ , then a similar equation holds for  $P(r, l, -b)$ . So  $Q = \frac{P(b|r)P(b|l)}{P(b)} + \frac{P(-b|r)P(-b|l)}{P(-b)}$

Contexts that occur primarily inside words (e.g. not at a syllable boundary) often restrict the adjacent context, violating conditional independence given  $-b$ . However, in these cases,  $P(b|r)$  and/or  $P(b|l)$  will be very low, so  $P(b|r, l)$  will be very low. So (correctly) no boundary will be marked.

Thus, we can compute  $P(b|r, l)$  from  $P(b|r)$ ,  $P(b|l)$ , and  $P(b)$ . A boundary is marked if  $P(b|r, l) \geq 0.5$ .

### 4.3 Estimating context probabilities

Estimation of  $P(b|r)$  and  $P(b|l)$  uses a simple ngram backoff algorithm. The details will be shown for  $P(b|l)$ .  $P(b|r)$  is similar.

Suppose for the moment that word boundaries are marked. The left context  $l$  might be very long and unusual. So we will estimate its statistics using a shorter lefthand neighborhood  $l'$ .  $P(b|l)$  is then estimated as the number of times  $l'$  occurs before a boundary, divided by the total number of times  $l'$  occurs in the corpus.

The suffix  $l'$  is chosen to be the longest suffix of  $l$  which occurs at least 10 times in the corpus, i.e. often enough for a reliable estimate in the presence

corpus	language	transcription	sm size	med size	lg size	pho/wd	wd/phr	hapax
BR87	English	dictionary	33K	–	–	2.9	3.4	31.7
Switchboard	English	dictionary	34K	409K	3086K	3.1	5.9	33.8
Switchboard	English	orthographic	34K	409K	3086K	[3.8]	5.9	34.2
Buckeye	English	dictionary	32K	290K	–	3.1	5.9	41.9
Buckeye	English	phonetic	32K	290K	–	2.9	5.9	66.0
Arabic	Arabic	dictionary	30K	405K	–	5.6	5.9	60.3
Spanish	Spanish	dictionary	37K	200K	–	3.7	8.4	49.1

Table 1: Key parameters for each test dataset include the language, transcription method, number of words (small, medium, large subsets), average phones per word, average words per phrase, and percent of word types that occur only once (hapax). Phones/word is replaced by characters/word for the orthographic corpus.

of noise.<sup>5</sup>  $l'$  may cross word boundaries and, if our position is near a pause, may contain the blank at the lefthand end of the phrase. The length of  $l'$  is limited to  $N_{max}$  characters to reduce overfitting.

Unfortunately, our input data has boundaries only at pauses (#). So applying this method to the raw input data produces estimates of  $P(\#|r)$  and  $P(\#|l)$ . Because phrase boundaries are not a representative selection of word boundaries,  $P(\#|r)$  and  $P(\#|l)$  are not good estimates of  $P(b|r)$  and  $P(b|l)$ . Moreover, initially, we don't know  $P(b)$ .

Therefore, WordEnds bootstraps the estimation using a binary model of the relationship between word and phrase boundaries. To a first approximation, an ngram occurs at the end of a phrase if and only if it can occur at the end of a word. Since the magnitude of  $P(\#, l)$  isn't helpful, we simply check whether it is zero and, accordingly, set  $P(b|l)$  to either zero or a constant, very high value.

In fact, real data contains phrase endings corrupted by disfluencies, foreign words, etc. So WordEnds actually sets  $P(b|l)$  high only if  $P(\#|l)$  is above a threshold (currently 0.003) chosen to reflect the expected amount of corruption.

In the equations from Section 4.2, if either  $P(b|r)$  or  $P(b|l)$  is zero, then  $P(b|r, l)$  is zero. If both values are very high, then  $Q$  is  $\frac{P(b|r)P(b|l)}{P(b)} + \epsilon$ , with  $\epsilon$  very small. So  $P(b|r, l)$  is close to 1. So, in the bootstrapping phase, the test for marking a boundary is independent of  $P(b)$  and reduces to testing whether  $P(\#|r)$  and  $P(\#|l)$  are both over threshold.

So, WordEnds estimates  $P(\#|r)$  and  $P(\#|l)$  from the input data, then uses this bootstrapping

<sup>5</sup>A single character is used if no suffix occurs 10 times.

method ( $N_{max} = 5$ )<sup>6</sup> to infer preliminary word boundaries. The preliminary boundaries are used to estimate  $P(b)$  and to re-estimate  $P(b|r)$  and  $P(b|l)$ , using  $N_{max} = 4$ . Final boundaries are then marked.

## 5 Mini-morph

In a full understanding system, output of the word segmenter would be passed to morphological and local syntactic processing. Because the segmenter is myopic, certain errors in its output would be easier to fix with the wider perspective available to this later processing. Because standard models of morphological learning don't address the interaction with word segmentation, WordEnds does a simple version of this repair process using a placeholder algorithm called Mini-morph.

Mini-morph fixes two types of defects in the segmentation. Short fragments are created when two nearby boundaries represent alternative reasonable segmentations rather than parts of a common segmentation. For example, "treestake" has potential boundaries both before and after the s. This issue was noted by Harrington et al. (1988) who used a list of known very short words to detect these cases. See also (Cairns et al., 1997). Also, surrounding words sometimes mislead WordEnds into undersegmenting a phone sequence which has an "obvious" analysis using well-established component words.

Mini-morph classifies each word in the segmentation as a fragment, a word that is reliable enough to use in subdividing other words, or unknown status.

<sup>6</sup>Values for  $N_{max}$  were chosen empirically. They could be adjusted for differences in entropy rate, but this is very similar across the datasets in this paper.

Because it has only a feeble model of morphology, Mini-morph has been designed to be cautious: most words are classified as unknown.

To classify a word, we compare its frequency  $w$  as a word in the segmentation to the frequencies  $p$  and  $s$  with which it occurs as a prefix and suffix of words in the segmentation (including itself). The word's fragment ratio  $f$  is  $\frac{2w}{p+s}$ .

Values of  $f$  are typically over 0.8 for freely occurring words, under 0.1 for fragments and strongly-attached affixes, and intermediate for clitics, some affixes, and words with restricted usage. However, most words haven't been seen enough times for  $f$  to be reliable. So a word is classified as a fragment if  $p + s \geq 1000$  and  $f \leq 0.2$ . It is classified as a reliable word if  $p + s \geq 50$  and  $f \geq 0.5$ .

To revise the input segmentation of the corpus, Mini-morph merges each fragment with an adjacent word if the newly-created merged word occurred at least 10 times in the input segmentation. When mergers with both adjacent words are possible, the algorithm alternates which to prefer. Each word is then subdivided into a sequence of reliable words, when possible. Because words are typically short and reliable words rare, a simple recursive algorithm is used, biased towards using shorter words.<sup>7</sup>

WordEnds calls Mini-morph twice, once to revise the preliminary segmentation produced by the bootstrapping phase and a second time to revise the final segmentation.

## 6 Test corpora

WordEnds was tested on a diverse set of seven corpora, summarized in Table 1. Notice that the Arabic dataset has much longer words than those used by previous authors. Subsets were extracted from the larger corpora, to control for training set size. Goldwater's algorithm, the best performing of previous methods, was also tested on the small versions.<sup>8</sup>

The first three corpora all use dictionary transcriptions with 1-character phone symbols. The Bernstein-Ratner (BR87) corpus was described above (Section 3.2). The Arabic corpus was created by removing punctuation and word boundaries from the Buckwalter version of the LDC's transcripts of

Gulf Arabic Conversational Telephone Speech (Apen, 2006). Filled pauses and foreign words were kept as is. Word fragments were kept, but the telltale hyphens were removed. The Spanish corpus was produced in a similar way from the Callhome Spanish dataset (Wheatley, 1996), removing all accents. Orthographic forms were used for words without pronunciations (e.g. foreign, fragments)

The other two English dictionary transcriptions were produced in a similar way from the Buckeye corpus (Pitt et al., 2005; Pitt et al., 2007) and Mississippi State's corrected version of the LDC's Switchboard transcripts (Godfrey and Holliman, 1994; Deshmukh et al., 1998). These use a "readable phonetic" version of arpabet. Each phone is represented with a 1–2 character code, chosen to look like English orthography and to ensure that character sequences decode uniquely into phone sequences. Buckeye does not provide dictionary pronunciations for word fragments, so these were transcribed as "X". Switchboard was also transcribed using standard English orthography.

The Buckeye corpus also provides an accurate phonetic transcription of its data, showing allophonic variation (e.g. glottal stop, dental/nasal flaps), segment deletions, quality shifts/uncertainty, and nasalization. Some words are "massively" reduced (Johnson, 2003), going well beyond standard phonological rules. We represented its 64 phones using codes with 1–3 characters.

## 7 Test results

Table 2 presents test results for the small corpora. The numbers for the four English dictionary and orthographic transcriptions are very similar. This confirms the finding of Batchelder (2002) that variations in transcription method have only minor impacts on segmenter performance. Performance seems to be largely determined by structural and lexical properties (e.g. word length, pause frequency).

For the English dictionary datasets, the primary overall evaluation numbers (BF and WF) for the two algorithms differ less than the variation created by tweaking parameters or re-running Goldwater's (randomized) algorithm. Both degrade similarly on the phonetic version of Buckeye. The most visible overall difference is speed. WordEnds processes

<sup>7</sup>Subdivision is done only once for each word type.

<sup>8</sup>It is too slow to run on the larger ones.

corpus	transcription	WordEnds					Goldwater				
		BP	BR	BF	WF	LF	BP	BR	BF	WF	LF
BR87	dictionary	<b>94.6</b>	73.7	82.9	70.7	36.6	89.2	<b>82.7</b>	<b>85.8</b>	<b>72.5</b>	<b>56.2</b>
Switchboard	dictionary	<b>91.3</b>	80.5	<b>85.5</b>	<b>72.0</b>	<b>37.4</b>	73.9	<b>93.5</b>	82.6	65.8	27.8
Switchboard	orthographic	<b>90.0</b>	75.5	<b>82.1</b>	<b>66.3</b>	<b>33.7</b>	73.1	<b>92.4</b>	81.6	63.6	28.4
Buckeye	dictionary	<b>89.7</b>	82.2	<b>85.8</b>	<b>72.3</b>	<b>37.4</b>	74.6	<b>94.8</b>	83.5	68.1	26.7
Buckeye	phonetic	<b>71.0</b>	64.1	<b>67.4</b>	<b>44.1</b>	<b>28.6</b>	49.6	<b>95.0</b>	65.1	35.4	12.8
Arab	dictionary	<b>88.1</b>	68.5	<b>77.1</b>	<b>56.6</b>	<b>40.4</b>	47.5	<b>97.4</b>	63.8	32.6	9.5
Spanish	dictionary	<b>89.3</b>	48.5	62.9	38.7	16.6	69.2	<b>92.8</b>	<b>79.3</b>	<b>57.9</b>	<b>17.0</b>

Table 2: Results for WordEnds and Goldwater on the small test corpora. See Section 2.3 for definitions of metrics.

corpus	transcription	medium w/out morph			medium			large		
		BF	WF	LF	BF	WF	LF	BF	WF	LF
Switchboard	dictionary	90.4	78.8	39.4	93.0	84.8	44.2	94.7	88.1	44.3
Switchboard	orthographic	89.6	77.4	37.3	91.6	81.8	41.1	94.1	87.0	41.1
Buckeye	dictionary	91.2	80.3	41.5	93.7	86.1	47.8	–	–	–
Buckeye	phonetic	72.1	48.4	27.1	75.0	54.2	28.2	–	–	–
Arab	dictionary	85.7	69.1	49.5	86.4	70.6	50.0	–	–	–
Spanish	dictionary	75.1	52.2	19.7	76.3	55.0	20.2	–	–	–

Table 3: Results for WordEnds on the medium and large datasets, also on the medium dataset without Mini-morph. See Table 1 for dataset sizes.

each small dataset in around 30-40 seconds. Goldwater requires around 2000 times as long: 14.5-32 hours, depending on the dataset.

However, WordEnds keeps affixes on words whereas Goldwater’s algorithm removes them. This creates a systematic difference in the balance between boundary recall and precision. It also causes Goldwater’s LF values to drop dramatically between the child-directed BR87 corpus and the adult-directed speech. For the same reason, WordEnds maintains good performance on the Arabic dataset, but Goldwater’s performance (especially LF) is much worse. It is quite likely that Goldwater’s algorithm is finding morphemes rather than words.

Datasets around 30K words are traditional for this task. However, a child learner has access to much more data, e.g. Weijer (1999) measured 1890 words per hour spoken near an infant. WordEnds performs much better when more data is available (Table 3). Numbers for even the harder datasets (Buckeye phonetic, Spanish) are starting to look promising. The Spanish results show that data with infrequent pauses can be handled in two very different ways: aggressive model-based segmentation (Gold-

water) or feeding more data to a more cautious segmenter (WordEnds).

The two calls to Mini-morph sometimes make almost no difference, e.g. on the Arabic data. But it can make large improvements, e.g. BF +6.9%, WF +10.5%, LF +5.8% on the BR corpus. Table 3 shows details for the medium datasets. Its contribution seems to diminish as the datasets get bigger, e.g. improvements of BF +4.7%, WF +9.3%, LF +3.7% on the small dictionary Switchboard corpus but only BF +1.3%, WF +3.3%, LF +3.4% on the large one.

## 8 Some specifics of performance

Examining specific mistakes confirms that WordEnds does not systematically remove affixes on English dictionary data. On the large Switchboard corpus, “-ed” is never removed from its stem and “-ing” is removed only 16 times. The Mini-morph post-processor misclassifies, and thus segments off, some affixes that are homophonous with free-standing words, such as “-en”/“in” and “-es”/“is”. A smarter model of morphology and local syntax could probably avoid this.

There is a visible difference between English “the” and the Arabic determiner “Al-”. The English determiner is almost always segmented off. From the medium-sized Switchboard corpus, only 434 lexical items are posited with “the” attached to a following word. Arabic “Al” is sometimes attached and sometimes segmented off. In the medium Arabic dataset, the correct and computed lexicons contain similar numbers of words starting with Al (4873 and 4608), but there is only partial overlap (2797 words). Some of this disagreement involves foreign language nouns, which the markup in the original corpus separates from the determiner.<sup>9</sup>

Mistakes on twenty specific items account for 24% of the errors on the large Switchboard corpus. The first two items, accounting for over 11% of the mistakes, involve splitting “uhhuh” and “umhum”. Most of the rest involve merging common collocations (e.g. “a lot”) or splitting common compounds that have a transparent analysis (e.g. “something”).

## 9 Discussion and conclusions

Performance of WordEnds is much stronger than previous reported results, including good results on Arabic and promising results on accurate phonetic transcriptions. This is partly due to good algorithm design and partly due to using more training data. This sets a much higher standard for models of child language acquisition and also suggests that it is not crazy to speculate about inserting such an algorithm into the speech recognition pipeline.

Performance would probably be improved by better models of morphology and/or phonology. An ngram model of morpheme sequences (e.g. like Goldwater uses) might avoid some of the mistakes mentioned in Section 8. Feature-based or gestural phonology (Browman and Goldstein, 1992) might help model segmental variation. Finite-state models (Belz, 2000) might be more compact. Prosody, stress, and other sub-phonemic cues might disambiguate some problem situations (Hockema, 2006; Rytting, 2007; Salverda et al., 2003).

However, it is not obvious which of these approaches will actually improve performance. Additional phonetic features may not be easy to detect

<sup>9</sup>The author does not read Arabic and, thus, is not in a position to explain why the annotators did this.

reliably, e.g. marking lexical stress in the presence of contrastive stress and utterance-final lengthening. The actual phonology of fast speech may not be quite what we expect, e.g. performance on the phonetic version of Buckeye was slightly **improved** by merging nasal flap with n, and dental flap with d and glottal stop. The sets of word initial and final segments may not form natural phonological classes, because they are partly determined by morphological and lexical constraints (Rytting, 2007).

Moreover, the strong performance from the basic segmental model makes it hard to rule out the possibility that high performance could be achieved, even on data with phonetic variation, by throwing enough training data at a simple segmental algorithm.

Finally, the role of child-directed speech needs to be examined more carefully. Child-directed speech displays helpful features such as shorter phrases and fewer reductions (Bernstein Ratner, 1996; van de Weijer, 1999). These features may make segmentation easier to learn, but the strong results presented here for adult-directed speech make it trickier to argue that this help is necessary for learning.

Moreover, it is not clear how learning to segment child-directed speech might make it easier to learn to segment speech directed at adults or older children. It’s possible that learning child-directed speech makes it easier to learn the basic principles of phonology, semantics, or higher-level linguistic structure. This might somehow feed back into learning segmentation. However, it’s also possible that its only *raison d’être* is social: enabling earlier communication between children and adults.

## Acknowledgments

Many thanks to the UIUC prosody group, Mitch Marcus, Cindy Fisher, and Sharon Goldwater.

## References

- Rie Kubota Ando and Lillian Lee. 2000. Mostly-Unsupervised Statistical Segmentation of Japanese. *Proc ANLP-NAACL 2000*:241–248.
- Appen Pty Ltd. 2006. Gulf Arabic Conversational Telephone Speech, Transcripts Linguistic Data Consortium, Philadelphia
- Eleanor Olds Batchelder 2002. Bootstrapping the lexicon: A computational model of infant speech segmentation. *Cognition* 83, pp. 167–206.

- Anja Belz 2000. Multi-Syllable Phonotactic Modelling. 5th ACL SIGPHON, pp. 46–56.
- Nan Bernstein Ratner. 1987. The phonology of parent child speech. In K. Nelson and A. Van Kleeck (Eds.), *Children’s Language: Vol 6*, Lawrence Erlbaum.
- Nan Bernstein Ratner 1996. From “Signal to Syntax”: But what is the Nature of the Signal? In James Morgan and Katherine Demuth (eds) *Signal to Syntax*, Lawrence Erlbaum, Mahwah, NJ.
- Michael R. Brent. 1999. An Efficient, Probabilistically Sound Algorithm for Segmentation and Word Discovery. *Machine Learning* 1999:71–105.
- Michael R. Brent and Timothy A. Cartwright. 1996. Distributional Regularity and Phonotactic Constraints are Useful for Segmentation *Cognition* 1996:93–125.
- C. P. Browman and L. Goldstein. 1992. Articulatory phonology: An overview. *Phonetica* 49:155–180.
- Paul Cairns, Richard Shillcock, Nick Chater, and Joe Levy. 1997. Bootstrapping Word Boundaries: A Bottom-up Corpus-based Approach to Speech Segmentation. *Cognitive Psychology*, 33:111–153.
- Morten Christiansen and Joseph Allen 1997. Coping with Variation in Speech Segmentation *GALA* 1997.
- Morten Christiansen, Joseph Allen, Mark Seidenberg. 1998. Learning to Segment Speech Using Multiple Cues: A Connectionist Model. *Language and Cognitive Processes* 12/2–3, pp. 221–268.
- Herbert H. Clark and Thomas Wasow. 1998. Repeating Words in Spontaneous Speech. *Cognitive Psychology* 37:201–242.
- N. Deshmukh, A. Ganapathiraju, A. Gleeson, J. Hamaker and J. Picone. 1998. Resegmentation of Switchboard. *Proc. Intern. Conf. on Spoken Language Processing*:1543–1546.
- Jean E. Fox Tree and Herbert H. Clark. 1997. Pronouncing “the” as “thee” to signal problems in speaking. *Cognition* 62(2):151–167.
- John J. Godfrey and Ed Holliman. 1993. Switchboard-1 Transcripts. Linguistic Data Consortium, Philadelphia, PA.
- Sharon Goldwater. 2006. Nonparametric Bayesian Models of Lexical Acquisition. Ph.D. thesis, Brown Univ.
- Jonathan Harrington, Gordon Watson, and Maggie Cooper. 1989. Word boundary detection in broad class and phoneme strings. *Computer Speech and Language* 3:367–382.
- Jonathan Harrington, Gordon Watson, and Maggie Cooper. 1988. Word Boundary Identification from Phoneme Sequence Constraints in Automatic Continuous Speech Recognition. *Coling* 1988, pp. 225–230.
- Stephen A. Hockema. 2006. Finding Words in Speech: An Investigation of American English. *Language Learning and Development*, 2(2):119–146.
- Keith Johnson 2003. Massive reduction in conversational American English. *Proc. of the Workshop on Spontaneous Speech: Data and Analysis*.
- Peter W. Jusczyk and Richard N. Aslin. 1995. Infants’ Detection of the Sound Patterns of Words in Fluent Speech. *Cognitive Psychology* 29(1)1–23.
- Lori F. Lamel and Victor W. Zue. 1984. Properties of Consonant Sequences within Words and Across Word Boundaries. *Proc. ICASSP* 1984:42.3.1–42.3.4.
- James M. McQueen. 1998. Segmentation of Continuous Speech Using Phonotactics. *Journal of Memory and Language* 39:21–46.
- Mark Pitt, Keith Johnson, Elizabeth Hume, Scott Kiesling, and William Raymond. 2005. The Buckeye Corpus of Conversational Speech: Labeling Conventions and a Test of Transcriber Reliability. *Speech Communication*, 45, 90–95.
- M. A. Pitt, L. Dilley, K. Johnson, S. Kiesling, W. Raymond., E. Hume, and E. Fosler-Lussier. 2007. Buckeye Corpus of Conversational Speech (2nd release) Department of Psychology, Ohio State University, Columbus, OH
- C. Anton Rytting 2004. Greek Word Segmentation using Minimal Information. *HLT-NAACL* 2004, pp. 78–85.
- C. Anton Rytting 2007. Preserving Subsegmental Variation in Modelling Word Segmentation. Ph.D. thesis, Ohio State, Columbus OH.
- J. R. Saffran. 2001 Words in a sea of sounds: The output of statistical learning. *Cognition* 81:149–169.
- Anne Pier Salverda, Delphine Dahan, and James M. McQueen. 2003. The role of prosodic boundaries in the resolution of lexical embedding in speech comprehension. *Cognition* 90:51–89.
- Amanda Seidl and Elizabeth K. Johnson. 2006. Infant Word Segmentation Revisited: Edge Alignment Facilitates Target Extraction. *Developmental Science* 9(6):565–573.
- W. J. Teahan, Y. Wen, R. McNab, I. H. Witten 2000 A compression-based algorithm for Chinese word segmentation. *Computational Linguistics* 26/3, pp. 375–393.
- Anand Venkataraman. 2001. A Statistical Model for Word Discovery in Transcribed Speech. *Computational Linguistics*, 27(3):351–372.
- A. Weber. 2000 Phonotactic and acoustic cues for word segmentation. *Proc. 6th Intern. Conf. on Spoken Language Processing*, Vol. 3: 782–785. pp
- Joost van de Weijer 1999. Language Input for Word Discovery. Ph.D. thesis, Katholieke Universiteit Nijmegen.
- Barbara Wheatley. 1996. CALLHOME Spanish Transcripts. Linguistic Data Consortium, Philadelphia.

# A Re-examination of Query Expansion Using Lexical Resources

Hui Fang

Department of Computer Science and Engineering  
The Ohio State University  
Columbus, OH, 43210  
hfang@cse.ohio-state.edu

## Abstract

Query expansion is an effective technique to improve the performance of information retrieval systems. Although hand-crafted lexical resources, such as WordNet, could provide more reliable related terms, previous studies showed that query expansion using only WordNet leads to very limited performance improvement. One of the main challenges is how to assign appropriate weights to expanded terms. In this paper, we re-examine this problem using recently proposed axiomatic approaches and find that, with appropriate term weighting strategy, we are able to exploit the information from lexical resources to significantly improve the retrieval performance. Our empirical results on six TREC collections show that query expansion using only hand-crafted lexical resources leads to significant performance improvement. The performance can be further improved if the proposed method is combined with query expansion using co-occurrence-based resources.

## 1 Introduction

Most information retrieval models (Salton et al., 1975; Fuhr, 1992; Ponte and Croft, 1998; Fang and Zhai, 2005) compute relevance scores based on matching of terms in queries and documents. Since various terms can be used to describe a same concept, it is unlikely for a user to use a query term that is exactly the same term as used in relevant documents. Clearly, such vocabulary gaps make the retrieval performance non-optimal. Query expansion (Voorhees, 1994; Mandala et al., 1999a; Fang and

Zhai, 2006; Qiu and Frei, 1993; Bai et al., 2005; Cao et al., 2005) is a commonly used strategy to bridge the vocabulary gaps by expanding original queries with related terms. Expanded terms are often selected from either co-occurrence-based thesauri (Qiu and Frei, 1993; Bai et al., 2005; Jing and Croft, 1994; Peat and Willett, 1991; Smeaton and van Rijsbergen, 1983; Fang and Zhai, 2006) or hand-crafted thesauri (Voorhees, 1994; Liu et al., 2004) or both (Cao et al., 2005; Mandala et al., 1999b).

Intuitively, compared with co-occurrence-based thesauri, hand-crafted thesauri, such as WordNet, could provide more reliable terms for query expansion. However, previous studies failed to show any significant gain in retrieval performance when queries are expanded with terms selected from WordNet (Voorhees, 1994; Stairmand, 1997). Although some researchers have shown that combining terms from both types of resources is effective, the benefit of query expansion using only manually created lexical resources remains unclear. The main challenge is how to assign appropriate weights to the expanded terms.

In this paper, we re-examine the problem of query expansion using lexical resources with the recently proposed axiomatic approaches (Fang and Zhai, 2006). The major advantage of axiomatic approaches in query expansion is to provide guidance on how to weight related terms based on a given term similarity function. In our previous study, a co-occurrence-based term similarity function was proposed and studied. In this paper, we study several term similarity functions that exploit various information from two lexical resources, i.e., WordNet

and dependency-thesaurus constructed by Lin (Lin, 1998), and then incorporate these similarity functions into the axiomatic retrieval framework. We conduct empirical experiments over several TREC standard collections to systematically evaluate the effectiveness of query expansion based on these similarity functions. Experiment results show that all the similarity functions improve the retrieval performance, although the performance improvement varies for different functions. We find that the most effective way to utilize the information from WordNet is to compute the term similarity based on the overlap of synset definitions. Using this similarity function in query expansion can significantly improve the retrieval performance. According to the retrieval performance, the proposed similarity function is significantly better than simple mutual information based similarity function, while it is comparable to the function proposed in (Fang and Zhai, 2006). Furthermore, we show that the retrieval performance can be further improved if the proposed similarity function is combined with the similarity function derived from co-occurrence-based resources.

The main contribution of this paper is to re-examine the problem of query expansion using lexical resources with a new approach. Unlike previous studies, we are able to show that query expansion using only manually created lexical resources can significantly improve the retrieval performance.

The rest of the paper is organized as follows. We discuss the related work in Section 2, and briefly review the studies of query expansion using axiomatic approaches in Section 3. We then present our study of using lexical resources, such as WordNet, for query expansion in Section 4, and discuss experiment results in Section 5. Finally, we conclude in Section 6.

## 2 Related Work

Although the use of WordNet in query expansion has been studied by various researchers, the improvement of retrieval performance is often limited. Voorhees (Voorhees, 1994) expanded queries using a combination of synonyms, hypernyms and hyponyms *manually* selected from WordNet, and achieved limited improvement (i.e., around  $-2\%$  to

$+2\%$ ) on short verbose queries. Stairmand (Stairmand, 1997) used WordNet for query expansion, but they concluded that the improvement was restricted by the coverage of the WordNet and no empirical results were reported.

More recent studies focused on combining the information from both co-occurrence-based and hand-crafted thesauri. Mandala et. al. (Mandala et al., 1999a; Mandala et al., 1999b) studied the problem in vector space model, and Cao et. al. (Cao et al., 2005) focused on extending language models. Although they were able to improve the performance, it remains unclear whether using only information from hand-crafted thesauri would help to improve the retrieval performance.

Another way to improve retrieval performance using WordNet is to disambiguate word senses. Voorhees (Voorhees, 1993) showed that using WordNet for word sense disambiguation degrade the retrieval performance. Liu et. al. (Liu et al., 2004) used WordNet for both sense disambiguation and query expansion and achieved reasonable performance improvement. However, the computational cost is high and the benefit of query expansion using only WordNet is unclear. Ruch et. al. (Ruch et al., 2006) studied the problem in the domain of biology literature and proposed an argumentative feedback approach, where expanded terms are selected from only sentences classified into one of four disjunct argumentative categories.

The goal of this paper is to study whether query expansion using only manually created lexical resources could lead to the performance improvement. The main contribution of our work is to show query expansion using only hand-crafted lexical resources is effective in the recently proposed axiomatic framework, which has not been shown in the previous studies.

## 3 Query Expansion in Axiomatic Retrieval Model

Axiomatic approaches have recently been proposed and studied to develop retrieval functions (Fang and Zhai, 2005; Fang and Zhai, 2006). The main idea is to search for a retrieval function that satisfies all the desirable retrieval constraints, i.e., axioms. The underlying assumption is that a retrieval function sat-



isfying all the constraints would perform well empirically. Unlike other retrieval models, axiomatic retrieval models directly model the relevance with term level retrieval constraints.

In (Fang and Zhai, 2005), several axiomatic retrieval functions have been derived based on a set of basic formalized retrieval constraints and an inductive definition of the retrieval function space. The derived retrieval functions are shown to perform as well as the existing retrieval functions with less parameter sensitivity. One of the components in the inductive definition is primitive weighting function, which assigns the retrieval score to a single term document  $\{d\}$  for a single term query  $\{q\}$  based on

$$S(\{q\}, \{d\}) = \begin{cases} \omega(q) & q = d \\ 0 & q \neq d \end{cases} \quad (1)$$

where  $\omega(q)$  is a term weighting function of  $q$ . A limitation of the primitive weighting function described in Equation 1 is that it can not bridge vocabulary gaps between documents and queries.

To overcome this limitation, in (Fang and Zhai, 2006), we proposed a set of semantic term matching constraints and modified the previously derived axiomatic functions to make them satisfy these additional constraints. In particular, the primitive weighting function is generalized as

$$S(\{q\}, \{d\}) = \omega(q) \times f(s(q, d)),$$

where  $s(q, d)$  is a semantic similarity function between two terms  $q$  and  $d$ , and  $f$  is a monotonically increasing function defined as

$$f(s(q, d)) = \begin{cases} 1 & q = d \\ \frac{s(q, d)}{s(q, q)} \times \beta & q \neq d \end{cases} \quad (2)$$

where  $\beta$  is a parameter that regulates the weighting of the original query terms and the semantically similar terms. We have shown that the proposed generalization can be implemented as a query expansion method. Specifically, the expanded terms are selected based on a term similarity function  $s$  and the weight of an expanded term  $t$  is determined by its term similarity with a query term  $q$ , i.e.,  $s(q, t)$ , as well as the weight of the query term, i.e.,  $\omega(q)$ . Note that the weight of an expanded term  $t$  is  $\omega(t)$  in traditional query expansion methods.

In our previous study (Fang and Zhai, 2006), term similarity function  $s$  is derived based on the mutual information of terms over collections that are constructed under the guidance of a set of term semantic similarity constraints. The focus of this paper is to study and compare several term similarity functions exploiting the information from lexical resources, and evaluate their effectiveness in the axiomatic retrieval models.

## 4 Term Similarity based on Lexical Resources

In this section, we discuss a set of term similarity functions that exploit the information stored in two lexical resources: WordNet (Miller, 1990) and dependency-based thesaurus (Lin, 1998).

The most commonly used lexical resource is WordNet (Miller, 1990), which is a hand-crafted lexical system developed at Princeton University. Words are organized into four taxonomies based on different parts of speech. Every node in the WordNet is a synset, i.e., a set of synonyms. The definition of a synset, which is referred to as *gloss*, is also provided. For a query term, all the synsets in which the term appears can be returned, along with the definition of the synsets. We now discuss six possible term similarity functions based on the information provided by WordNet.

Since the definition provides valuable information about the semantic meaning of a term, we can use the definitions of the terms to measure their semantic similarity. The more common words the definitions of two terms have, the more similar these terms are (Banerjee and Pedersen, 2005). Thus, we can compute the term semantic similarity based on synset definitions in the following way:

$$s_{def}(t_1, t_2) = \frac{|D(t_1) \cap D(t_2)|}{|D(t_1) \cup D(t_2)|},$$

where  $D(t)$  is the concatenation of the definitions for all the synsets containing term  $t$  and  $|D|$  is the number of words of the set  $D$ .

Within a taxonomy, synsets are organized by their lexical relations. Thus, given a term, related terms can be found in the synsets related to the synsets containing the term. In this paper, we consider the following five word relations.

- **Synonym(Syn)**: X and Y are synonyms if they are interchangeable in some context.
- **Hypernym(Hyper)**: Y is a hypernym of X if X is a (kind of) Y.
- **Hyponym(Hypo)**: X is a hyponym of Y if X is a (kind of) Y.
- **Holonym(Holo)**: Y is a holonym of X if X is a part of Y.
- **Meronym(Mero)**: X is a meronym of Y if X is a part of Y.

Since these relations are binary, we define the term similarity functions based on these relations in the following way.

$$s_R(t_1, t_2) = \begin{cases} \alpha_R & t_1 \in T_R(t_2) \\ 0 & t_1 \notin T_R(t_2) \end{cases}$$

where  $R \in \{syn, hyper, hypo, holo, mero\}$ ,  $T_R(t)$  is a set of words that are related to term  $t$  based on the relation  $R$ , and  $\alpha$ s are non-zero parameters to control the similarity between terms based on different relations. However, since the similarity values for all term pairs are same, the values of these parameters can be ignored when we use Equation 2 in query expansion.

Another lexical resource we study in the paper is the dependency-based thesaurus provided by Lin<sup>1</sup> (Lin, 1998). The thesaurus provides term similarities that are automatically computed based on dependency relationships extracted from a parsed corpus. We define a similarity function that can utilize this thesaurus as follows:

$$s_{Lin}(t_1, t_2) = \begin{cases} L(t_1, t_2) & (t_1, t_2) \in TP_{Lin} \\ 0 & (t_1, t_2) \notin TP_{Lin} \end{cases}$$

where  $L(t_1, t_2)$  is the similarity of terms stored in the dependency-based thesaurus and  $TP_{Lin}$  is a set of all the term pairs stored in the thesaurus. The similarity of two terms would be assigned to zero if we can not find the term pair in the thesaurus.

Since all the similarity functions discussed above capture different perspectives of term relations, we

propose a simple strategy to combine these similarity functions so that the similarity of a term pair is the highest similarity value of these two terms of all the above similarity functions, which is shown as follows.

$$s_{combined}(t_1, t_2) = \max_{R \in Rset}(s_R(t_1, t_2)),$$

where

$$Rset = \{def, syn, hyper, hypo, holo, mero, Lin\}.$$

In summary, we have discussed eight possible similarity functions that exploit the information from the lexical resources. We then incorporate these similarity functions into the axiomatic retrieval models based on Equation 2, and perform query expansion based on the procedure described in Section 3. The empirical results are reported in Section 5.

## 5 Experiments

In this section, we experimentally evaluate the effectiveness of query expansion with the term similarity functions discussed in Section 4 in the axiomatic framework. Experiment results show that the similarity function based on synset definitions is most effective. By incorporating this similarity function into the axiomatic retrieval models, we show that query expansion using the information from only WordNet can lead to significant improvement of retrieval performance, which has not been shown in the previous studies (Voorhees, 1994; Stairmand, 1997).

### 5.1 Experiment Design

We conduct three sets of experiments. First, we compare the effectiveness of term similarity functions discussed in Section 4 in the context of query expansion. Second, we compare the best one with the term similarity functions derived from co-occurrence-based resources. Finally, we study whether the combination of term similarity functions from different resources can further improve the performance.

All experiments are conducted over six TREC collections: ap88-89, doe, fr88-89, wt2g, trec7 and trec8. Table 1 shows some statistics of the collections, including the description, the collection size,

<sup>1</sup>Available at <http://www.cs.ualberta.ca/~lindek/downloads.htm>

Table 1: Statistics of Test Collections

Collection	Description	Size	# Voc.	# Doc.	#query
ap88-89	news articles	491MB	361K	165K	150
doe	technical reports	184MB	163K	226K	35
fr88-89	government documents	469MB	204K	204K	42
trec7	ad hoc data	2GB	908K	528K	50
trec8	ad hoc data	2GB	908K	528K	50
wt2g	web collections	2GB	1968K	247K	50

the vocabulary size, the number of documents and the number of queries. The preprocessing only involves stemming with Porter’s stemmer.

We use WordNet 3.0 <sup>2</sup>, Lemur Toolkit <sup>3</sup> and TrecWN library <sup>4</sup> in experiments. The results are evaluated with both MAP (mean average precision) and gMAP (geometric mean average precision) (Voorhees, 2005), which emphasizes the performance of difficulty queries.

There is one parameter  $\beta$  in the query expansion method presented in Section 3. We tune the value of  $\beta$  and report the best performance. The parameter sensitivity is similar to the observations described in (Fang and Zhai, 2006) and will not be discussed in this paper. In all the result tables, ‡ and † indicate that the performance difference is statistically significant according to Wilcoxon signed rank test at the level of 0.05 and 0.1 respectively.

We now explain the notations of different methods. *BL* is the baseline method without query expansion. In this paper, we use the best performing function derived in axiomatic retrieval models, i.e, F2-EXP in (Fang and Zhai, 2005) with a fixed parameter value ( $b = 0.5$ ).  $QE_X$  is the query expansion method with term similarity function  $s_X$ , where  $X$  could be *Def.*, *Syn.*, *Hyper.*, *Hypo.*, *Mero.*, *Holo.*, *Lin* and *Combined*.

Furthermore, we examine the query expansion method using co-occurrence-based resources. In particular, we evaluate the retrieval performance using the following two similarity functions:  $s_{MIBL}$  and  $s_{MIImp}$ . Both functions are based on the mutual information of terms in a set of documents.  $s_{MIBL}$  uses the collection itself to compute the mutual information, while  $s_{MIImp}$  uses the working sets con-

structed based on several constraints (Fang and Zhai, 2006). The mutual information of two terms  $t_1$  and  $t_2$  in collection  $C$  is computed as follow (van Rijsbergen, 1979):

$$I(X_{t_1}, X_{t_2}) = \sum p(X_{t_1}, X_{t_2}) \log \frac{p(X_{t_1}, X_{t_2})}{p(X_{t_1})p(X_{t_2})}$$

$X_{t_i}$  is a binary random variable corresponding to the presence/absence of term  $t_i$  in each document of collection  $C$ .

## 5.2 Effectiveness of Lexical Resources

We first compare the retrieval performance of query expansion with different similarity functions using short keyword (i.e., title-only) queries, because query expansion techniques are often more effective for shorter queries (Voorhees, 1994; Fang and Zhai, 2006). The results are presented in Table 2. It is clear that query expansion with these functions can improve the retrieval performance, although the performance gains achieved by different functions vary a lot. In particular, we make the following observations.

First, the similarity function based on synset definitions is the most effective one.  $QE_{def}$  significantly improves the retrieval performance for all the data sets. For example, in trec7, it improves the performance from 0.186 to 0.216. As far as we know, none of the previous studies showed such significant performance improvement by using only WordNet as query expansion resource.

Second, the similarity functions based on term relations are less effective compared with definition-based similarity function. We think that the worse performance is related to the following two reasons: (1) The similarity functions based on relations are binary, which is not a good way to model term similarities. (2) The relations are limited by the part

<sup>2</sup><http://wordnet.princeton.edu/>

<sup>3</sup><http://www.lemurproject.org/>

<sup>4</sup><http://l2r.cs.uiuc.edu/cogcomp/software.php>

Table 2: Performance of query expansion using lexical resources (short keyword queries)

	trec7		trec8		wt2g	
	MAP	gMAP	MAP	gMAP	MAP	gMAP
<i>BL</i>	0.186	0.083	0.250	0.147	0.282	0.188
$QE_{def}$	<b>0.216<math>\ddagger</math></b>	<b>0.105<math>\ddagger</math></b>	<b>0.266<math>\ddagger</math></b>	<b>0.164<math>\ddagger</math></b>	<b>0.301<math>\ddagger</math></b>	<b>0.210<math>\ddagger</math></b>
	<b>(+16%)</b>	<b>(+27%)</b>	<b>(+6.4%)</b>	<b>(+12%)</b>	<b>(+6.7%)</b>	<b>(+12%)</b>
$QE_{syn}$	0.194	0.085 $\ddagger$	0.252 $\ddagger$	0.150 $\ddagger$	0.287 $\ddagger$	0.194 $\ddagger$
	(+4.3%)	(+2.4%)	(+0.8%)	(+2.0%)	(+1.8%)	(+3.2%)
$QE_{hyper}$	0.186	0.086	0.250	0.152	0.286 $\ddagger$	0.192 $\ddagger$
	(0%)	(+3.6%)	(0%)	(+3.4%)	(+1.4%)	(+2.1%)
$QE_{hypo}$	0.186 $\ddagger$	0.085 $\ddagger$	0.250	0.147	0.282 $\ddagger$	0.190
	(0%)	(+2.4%)	(0%)	(0%)	(0%)	(+1.1%)
$QE_{mero}$	0.187 $\ddagger$	0.084 $\ddagger$	0.250	0.147	0.282	0.189
	(+0.5%)	(+1.2%)	(0%)	(0%)	(0%)	(+0.5%)
$QE_{holo}$	0.191 $\ddagger$	0.085 $\ddagger$	0.250	0.147	0.282	0.188
	(+2.7%)	(+2.4%)	(0%)	(0%)	(0%)	(0%)
$QE_{Lin}$	0.193 $\ddagger$	0.092 $\ddagger$	0.256 $\ddagger$	0.156 $\ddagger$	0.290 $\ddagger$	0.200 $\ddagger$
	(+3.7%)	(+11%)	(+2.4%)	(+6.1%)	(+2.8%)	(+6.4%)
$QE_{Combined}$	0.214 $\ddagger$	0.104 $\ddagger$	<b>0.267<math>\ddagger</math></b>	<b>0.165<math>\ddagger</math></b>	0.300 $\ddagger$	0.208 $\ddagger$
	(+15%)	(+25%)	(+6.8%)	(+12%)	(+6.4%)	(+10.5%)
	ap88-89		doe		fr88-89	
	MAP	gMAP	MAP	gMAP	MAP	gMAP
<i>BL</i>	0.220	0.074	0.174	0.069	0.222	0.062
$QE_{def}$	<b>0.254<math>\ddagger</math></b>	<b>0.088<math>\ddagger</math></b>	<b>0.181<math>\ddagger</math></b>	<b>0.075<math>\ddagger</math></b>	<b>0.225<math>\ddagger</math></b>	<b>0.067<math>\ddagger</math></b>
	<b>(+15%)</b>	<b>(+19%)</b>	<b>(+4%)</b>	<b>(+10%)</b>	<b>(+1.4%)</b>	<b>(+8.1%)</b>
$QE_{syn}$	0.222 $\ddagger$	0.077 $\ddagger$	0.174	0.074	0.222	0.065
	(+0.9%)	(+4.1%)	(0%)	(+7.3%)	(0%)	(+4.8%)
$QE_{hyper}$	0.222 $\ddagger$	0.074	0.175	0.070	0.222	0.062
	(+0.9%)	(0%)	(+0.5%)	(+1.5%)	(0%)	(0%)
$QE_{hypo}$	0.222 $\ddagger$	0.076 $\ddagger$	0.176 $\ddagger$	0.073 $\ddagger$	0.222	0.062
	(+0.9%)	(+2.7%)	(+1.1%)	(+5.8%)	(0%)	(0%)
$QE_{mero}$	0.221	0.074 $\ddagger$	0.174 $\ddagger$	0.070 $\ddagger$	0.222	0.062
	(+0.45%)	(0%)	(0%)	(+1.5%)	(0%)	(0%)
$QE_{holo}$	0.221	0.076	0.177 $\ddagger$	0.073	0.222	0.062
	(+0.45%)	(+2.7%)	(+1.7%)	(+5.8%)	(0%)	(0%)
$QE_{Lin}$	0.245 $\ddagger$	0.082 $\ddagger$	0.178	0.073	0.222	0.067 $\ddagger$
	(+11%)	(+11%)	(+2.3%)	(+5.8%)	(0%)	(+8.1%)
$QE_{Combined}$	<b>0.254<math>\ddagger</math></b>	0.085 $\ddagger$	0.179 $\ddagger$	0.074 $\ddagger$	0.223 $\ddagger$	0.065
	(+15%)	(+12%)	(+2.9%)	(+7.3%)	(+0.5%)	(+4.3%)

Table 3: Performance comparison of hand-crafted and co-occurrence-based thesauri (short keyword queries)

Data	MAP			gMAP		
	$QE_{def}$	$QE_{MIBL}$	$QE_{MIImp}$	$QE_{def}$	$QE_{MIBL}$	$QE_{MIImp}$
<b>ap88-89</b>	0.254	0.233‡	0.265‡	0.088	0.081‡	0.089‡
<b>doe</b>	0.181	0.175†	0.183	0.075	0.071†	0.078
<b>fr88-89</b>	0.225	0.222‡	0.227†	0.067	0.063	0.071‡
<b>trec7</b>	0.216	0.195‡	0.236‡	0.105	0.089‡	0.097
<b>trec8</b>	0.266	0.250‡	0.278	0.164	0.148‡	0.172
<b>wt2g</b>	0.301	0.311	0.320‡	0.210	0.218	0.219‡

of speech of the terms, because two terms in WordNet are related only when they have the same part of speech tags. However, definition-based similarity function does not have such a limitation.

Third, the similarity function based on Lin’s thesaurus is more effective than those based on term relations from the WordNet, while it is less effective compared with the definition-based similarity function, which might be caused by its smaller coverage.

Finally, combining different WordNet-based similarity functions does not help, which may indicate that the expanded terms selected by different functions are overlapped.

### 5.3 Comparison with Co-occurrence-based Resources

As shown in Table 2, the similarity function based on synset definitions, i.e.,  $s_{def}$ , is most effective. We now compare the retrieval performance of using this similarity function with that of using the mutual information based functions, i.e.,  $s_{MIBL}$  and  $s_{MIImp}$ . The experiments are conducted over two types of queries, i.e. short keyword (keyword title) and short verbose (one sentence description) queries.

The results for short keyword queries are shown in Table 3. The retrieval performance of query expansion based on  $s_{def}$  is significantly better than that based on  $s_{MIBL}$  on almost all the data sets, while it is slightly worse than that based on  $s_{MIImp}$  on some data sets. We can make the similar observation from the results for short verbose queries as shown in Table 4. One advantage of  $s_{def}$  over  $s_{MIImp}$  is the computational cost, because  $s_{def}$  can be computed offline in advance while  $s_{MIImp}$  has to be computed online from query-dependent working sets which takes much more time. The low computa-

tional cost and high retrieval performance make  $s_{def}$  more attractive in the real world applications.

### 5.4 Additive Effect

Since both types of similarity functions are able to improve retrieval performance, we now study whether combining them could lead to better performance. Table 5 shows the retrieval performance of combining both types of similarity functions for short keyword queries. The results for short verbose queries are similar. Clearly, combining the similarity functions from different resources could further improve the performance.

## 6 Conclusions

Query expansion is an effective technique in information retrieval to improve the retrieval performance, because it often can bridge the vocabulary gaps between queries and documents. Intuitively, hand-crafted thesaurus could provide reliable related terms, which would help improve the performance. However, none of the previous studies is able to show significant performance improvement through query expansion using information only from manually created lexical resources.

In this paper, we re-examine the problem of query expansion using lexical resources in recently proposed axiomatic framework and find that we are able to significantly improve retrieval performance through query expansion using only hand-crafted lexical resources. In particular, we first study a few term similarity functions exploiting the information from two lexical resources: WordNet and dependency-based thesaurus created by Lin. We then incorporate the similarity functions with the query expansion method in the axiomatic retrieval

Table 4: Performance Comparison (MAP, short verbose queries)

Data	$BL$	$QE_{def}$	$QE_{MIBL}$	$QE_{MIImp}$
<b>ap88-89</b>	0.181	0.220 $\ddagger$ (21.5%)	0.205 $\ddagger$ (13.3%)	0.230 $\ddagger$ (27.1%)
<b>doe</b>	0.109	0.121 $\ddagger$ (11%)	0.119 (9.17%)	0.117 (7.34%)
<b>fr88-89</b>	0.146	0.164 $\ddagger$ (12.3%)	0.162 $\ddagger$ (11%)	0.164 $\ddagger$ (12.3%)
<b>trec7</b>	0.184	0.209 $\ddagger$ (13.6%)	0.196 (6.52%)	0.224 $\ddagger$ (21.7%)
<b>trec8</b>	0.234	0.238 $\ddagger$ (1.71%)	0.235 (0.4%)	0.243 $\ddagger$ (3.85%)
<b>wt2g</b>	0.266	0.276 (3.76%)	0.276 $\ddagger$ (3.76%)	0.282 $\ddagger$ (6.02%)

Table 5: Additive Effect (MAP, short keyword queries)

	<b>ap88-89</b>	<b>doe</b>	<b>fr88-89</b>	<b>trec7</b>	<b>trec8</b>	<b>wt2g</b>
$QE_{MIBL}$	0.233	0.175	0.222	0.195	0.250	0.311
$QE_{def+MIBL}$	0.257 $\ddagger$	0.183 $\ddagger$	0.225 $\ddagger$	0.217 $\ddagger$	0.267 $\ddagger$	0.320 $\ddagger$
$QE_{MIImp}$	0.265	0.183	0.227	0.236	0.278	0.320
$QE_{def+MIImp}$	<b>0.269<math>\ddagger</math></b>	<b>0.187</b>	<b>0.232<math>\ddagger</math></b>	<b>0.237<math>\ddagger</math></b>	<b>0.280<math>\ddagger</math></b>	<b>0.322<math>\ddagger</math></b>

models. Systematical experiments have been conducted over six standard TREC collections and show promising results. All the proposed similarity functions improve the retrieval performance, although the degree of improvement varies for different similarity functions. Among all the functions, the one based on synset definition is most effective and is able to significantly and consistently improve retrieval performance for all the data sets. This similarity function is also compared with some similarity functions using mutual information. Furthermore, experiment results show that combining similarity functions from different resources could further improve the performance.

Unlike previous studies, we are able to show that query expansion using only manually created thesauri can lead to significant performance improvement. The main reason is that the axiomatic approach provides guidance on how to appropriately assign weights to expanded terms.

There are many interesting future research directions based on this work. First, we will study the same problem in some specialized domain, such as biology literature, to see whether the proposed approach could be generalized to the new domain. Second, the fact that using axiomatic approaches to incorporate linguistic information can improve retrieval performance is encouraging. We plan to extend the axiomatic approach to incorporate more linguistic information, such as phrases and word

senses, into retrieval models to further improve the performance.

## Acknowledgments

We thank ChengXiang Zhai, Dan Roth, Rodrigo de Salvo Braz for valuable discussions. We also thank three anonymous reviewers for their useful comments.

## References

- J. Bai, D. Song, P. Bruza, J. Nie, and G. Cao. 2005. Query expansion using term relationships in language models for information retrieval. In *Fourteenth International Conference on Information and Knowledge Management (CIKM 2005)*.
- S. Banerjee and T. Pedersen. 2005. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*.
- G. Cao, J. Nie, and J. Bai. 2005. Integrating word relationships into language models. In *Proceedings of the 2005 ACM SIGIR Conference on Research and Development in Information Retrieval*.
- H. Fang and C. Zhai. 2005. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 2005 ACM SIGIR Conference on Research and Development in Information Retrieval*.
- H. Fang and C. Zhai. 2006. Semantic term matching in axiomatic approaches to information retrieval. In *Proceedings of the 2006 ACM SIGIR Conference on Research and Development in Information Retrieval*.

- N. Fuhr. 1992. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255.
- Y. Jing and W. Bruce Croft. 1994. An association thesaurus for information retrieval. In *Proceedings of RIAO*.
- D. Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of International Conference on Machine Learning (ICML)*.
- S. Liu, F. Liu, C. Yu, and W. Meng. 2004. An effective approach to document retrieval via utilizing wordnet and recognizing phrases. In *Proceedings of the 2004 ACM SIGIR Conference on Research and Development in Information Retrieval*.
- R. Mandala, T. Tokunaga, and H. Tanaka. 1999a. Ad hoc retrieval experiments using wordnet and automatically constructed thesauri. In *Proceedings of the seventh Text REtrieval Conference (TREC7)*.
- R. Mandala, T. Tokunaga, and H. Tanaka. 1999b. Combining multiple evidence from different types of thesaurus for query expansion. In *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*.
- G. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4).
- H. J. Peat and P. Willett. 1991. The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science*, 42(5):378–383.
- J. Ponte and W. B. Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR'98*, pages 275–281.
- Y. Qiu and H.P. Frei. 1993. Concept based query expansion. In *Proceedings of the 1993 ACM SIGIR Conference on Research and Development in Information Retrieval*.
- P. Ruch, I. Tbahriti, J. Gobeill, and A. R. Aronson. 2006. Argumentative feedback: A linguistically-motivated term expansion for information retrieval. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 675–682.
- G. Salton, C. S. Yang, and C. T. Yu. 1975. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44, Jan-Feb.
- A. F. Smeaton and C. J. van Rijsbergen. 1983. The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal*, 26(3):239–246.
- M. A. Stairmand. 1997. Textual context analysis for information retrieval. In *Proceedings of the 1997 ACM SIGIR Conference on Research and Development in Information Retrieval*.
- C. J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths.
- E. M. Voorhees. 1993. Using wordnet to disambiguate word sense for text retrieval. In *Proceedings of the 1993 ACM SIGIR Conference on Research and Development in Information Retrieval*.
- E. M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 1994 ACM SIGIR Conference on Research and Development in Information Retrieval*.
- E. M. Voorhees. 2005. Overview of the trec 2005 robust retrieval track. In *Notebook of the Thirteenth Text REtrieval Conference (TREC2005)*.

# Selecting Query Term Alterations for Web Search by Exploiting Query Contexts

**Guihong Cao**

Dept. of Computer Science and  
Operations Research  
University of Montreal, Canada  
caogui@iro.umontreal.ca

**Stephen Robertson**

Microsoft Research at  
Cambridge  
Cambridge, UK  
ser@microsoft.com

**Jian-Yun Nie**

Dept. of Computer Science and  
Operations Research  
University of Montreal, Canada  
nie@iro.umontreal.ca

## Abstract

Query expansion by word alterations (alternative forms of a word) is often used in Web search to replace word stemming. This allows users to specify particular word forms in a query. However, if many alterations are added, query traffic will be greatly increased. In this paper, we propose methods to select only a few useful word alterations for query expansion. The selection is made according to the appropriateness of the alteration to the query context (using a bigram language model), or according to its expected impact on the retrieval effectiveness (using a regression model). Our experiments on two TREC collections will show that both methods only select a few expansion terms, but the retrieval effectiveness can be improved significantly.

## 1 Introduction

Word stemming is a basic NLP technique used in most of Information Retrieval (IR) systems. It transforms words into their root forms so as to increase the chance to match similar words/terms that are morphological variants. For example, with stemming, “controlling” can match “controlled” because both have the same root “control”. Most stemmers, such as the Porter stemmer (Porter, 1980) and Krovetz stemmer (Krovetz, 1993), deal with stemming by stripping word suffixes according to a set of morphological rules. Rule-based approaches are intuitive and easy to implement. However, while in general, most words can be stemmed correctly; there is often erroneous stemming that unifies unrelated words. For instance,

“jobs” is stemmed to “job” in both “find jobs in Apple” and “Steve Jobs at Apple”. This is particularly problematic in Web search, where users often use special or new words in their queries. A standard stemmer such as Porter’s will wrongly stem them.

To better determine stemming rules, Xu and Croft (1998) propose a selective stemming method based on corpus analysis. They refine the Porter stemmer by means of word clustering: words are first clustered according to their co-occurrences in the text collection. Only word variants belonging to the same cluster will be conflated.

Despite this improvement, the basic idea of word stemming is to transform words in both documents and queries to a standard form. Once this is done, there is no means for users to require a specific word form in a query – the word form will be automatically transformed, otherwise, it will not match documents. This approach does not seem to be appropriate for Web search, where users often specify particular word forms in their queries. An example of this is a quoted query such as “Steve Jobs”, or “US Policy”. If documents are stemmed, many pages about job offerings or US police may be returned (“policy” conflates with “police” in Porter stemmer). Another drawback of stemming is that it usually enhances recall, but may hurt precision (Kraaij and Pohlmann, 1996). However, general Web search is basically a precision-oriented task.

One alternative approach to word stemming is to do query expansion at query time. The original query terms are expanded by their related forms having the same root. All expansions can be combined by the Boolean operator “OR”. For example,



the query “*controlling acid rain*” can be expanded to “(*control OR controlling OR controller OR controlled OR controls*) (*acid OR acidic OR acidify*) (*rain OR raining OR rained OR rains*)”. We will call each such expansion term an *alteration* to the original query term. Once a set of possible alterations is determined, the simplest approach to perform expansion is to add all possible alterations. We call this approach *Naive Expansion*. One can easily show that stemming at indexing time is equivalent to *Naive Expansion* at retrieval time. This approach has been adopted by most commercial search engines (Peng et al., 2007). However, the expansion approaches proposed previously can have several serious problems: First, they usually do not consider expansion ambiguity – each query term is usually expanded independently. However, some expansion terms may not be appropriate. The case of “Steve Jobs” is one such example, for which the word “job” can be proposed as an expansion term. Second, as each query term may have several alterations, the naïve approach using all the alterations will create a very long query. As a consequence, query traffic (the time required for the evaluation of a query) is greatly increased. Query traffic is a critical problem, as each search engine serves millions of users at the same time. It is important to limit the query traffic as much as possible.

In practice, we can observe that some word alterations are irrelevant and undesirable (as in the “Steve Jobs” case), and some other alterations have little impact on the retrieval effectiveness (for example, if we expand a word by a rarely used word form). In this study, we will address these two problems. Our goal is to select only appropriate word alterations to be used in query expansion. This is done for two purposes: On the one hand, we want to limit query traffic as much as possible when query expansion is performed. On the other hand, we also want to remove irrelevant expansion terms so that fewer irrelevant documents will be retrieved, thereby improve the retrieval effectiveness.

To deal with the two problems we mentioned above, we will propose two methods to select alterations. In the first method, we make use of the query context to select only the alterations that fit the query. The query context is modeled by a bigram language model. To reduce query traffic, we select only one alteration for each query term,

which is the most coherent with the bigram model. We call this model *Bigram Expansion*. Despite the fact that this method adds far fewer expansion terms than the naïve expansion, our experiments will show that we can achieve comparable or even better retrieval effectiveness.

Both the Naive Expansion and the Bigram Expansion determine word alterations solely according to general knowledge about the language (bigram model or morphological rules), and no consideration about the possible effect of the expansion term is made. In practice, some alterations will have virtually no impact on retrieval effectiveness. They can be ignored. Therefore, in our second method, we will try to predict whether an alteration will have some positive impact on retrieval effectiveness. Only the alterations with positive impact will be retained. In this paper, we will use a regression model to predict the impact on retrieval effectiveness. Compared to the bigram expansion method, the regression method results in even fewer alterations, but experiments show that the retrieval effectiveness is even better.

Experiments will be conducted on two TREC collections, Gov2 data for Web Track and TREC6&7&8 for ad-hoc retrieval. The results show that the two methods we propose both outperform the original queries significantly with less than two alterations per query on average. Compared to the *Naive Expansion* method, the two methods can perform at least equally well, while query traffic is dramatically reduced.

In the following section, we provide a brief review of related work. Section 3 shows how to generate alteration candidates using a similar approach to Xu and Croft’s corpus analysis (1998). In section 4 and 5, we describe the Bigram Expansion method and Regression method respectively. Section 6 presents some experiments on TREC benchmarks to evaluate our methods. Section 7 concludes this paper and suggests some avenues for future work.

## 2 Related Work

Many stemmers have been implemented and used as standard processing in IR. Among them, the Porter stemmer (Porter, 1980) is the most widely used. It strips term suffixes step-by-step according to a set of morphological rules. However, the Porter stemmer sometimes wrongly transforms a term into an unrelated root. For example, it will unify

“news” and “new”, “execute” and “executive”. On the other hand, it may miss some confluences, such as “mice” and “mouse”, “europe” and “european”. Krovetz (1993) developed another stemmer, which uses a machine-readable dictionary, to improve the Porter stemmer. It avoids some of the Porter stemmer’s wrong stripping, but does not produce consistent improvement in IR experiments.

Both stemmers use generic rules for English to strip each word in isolation. In practice, the required stemming may vary from one text collection to another. Therefore, attempts have been made to use corpus analysis to improve existing rule-based stemmers. Xu and Croft (1998) create equivalence clusters of words which are morphologically similar and occur in similar contexts.

As we stated earlier, the stemming-based IR approaches are not well suited to Web search. Query expansion has been used as an alternative (Peng et al. 2007). To limit the number of expansion terms, and thus the query traffic, Peng et al. only use alterations for some of the query words: They segment each query into phrases and only the head word in each phrase is expanded. The assumptions are: 1) Queries issued in Web search often consist of noun phrases. 2) Only the head word in the noun phrase varies in form and needs to be expanded. However, both assumptions may be questionable. Their experiments did not show that the two assumptions hold.

Stemming is related to query expansion or query reformulation (Jones et al., 2006; Anick, 2003; Xu and Croft, 1996), although the latter is not limited to word variants. If the expansion terms used are those that are variant forms of a word, then query expansion can produce the same effect as word stemming. However, if we add all possible word alterations, query expansion/reformulation will run the risk of adding many unrelated terms to the original query, which may result in both heavy traffic and topic drift. Therefore, we need a way to select the most appropriate expansion terms. In (Peng et al. 2007), a bigram language model is used to determine the alteration of the head word that best fits the query. In this paper, one of the proposed methods will also use a bigram language model of the query to determine the appropriate alteration candidates. However, in our approach, alterations are not limited to head words. In addition, we will also propose a supervised learning

method to predict if an alteration will have a positive impact on retrieval effectiveness. To our knowledge, no previous method uses the same approach.

In the following sections, we will describe our approach, which consists of two steps: the generation of alteration candidates, and the selection of appropriate alterations for a query. The first step is query-independent using corpus analysis, while the second step is query-dependent. The selected word alterations will be *OR*-ed with the original query words.

### 3 Generating Alteration Candidates

Our method to generate alteration candidates can be described as follows. First, we do word clustering using a Porter stemmer. All words in the vocabulary sharing the same root form are grouped together. Then we do corpus analysis to filter out the words which are clustered incorrectly, according to word distributional similarity, following (Xu and Croft, 1998; Lin 1998). The rationale behind this is that words sharing the same meaning tend to occur in the same contexts.

The context of each word in the vocabulary is represented by a vector containing the frequencies of the context words which co-occur with the word within a predefined window in a training corpus. The window size is set empirically at 3 words and the training corpus is about 1/10 of the GOV2 corpus (see section 5 for details about the collection). Similarity is measured by the cosine distance between two vectors. For each word, we select at most 5 similar words as alteration candidates.

In the next sections, we will further consider ways to select appropriate alterations according to the query.

### 4 Bigram Expansion Model for Alteration Selection

In this section, we try to select the most suitable alterations according to the query context. The query context is modeled by a bigram language model as in (Peng et al. 2007).

Given a query described by a sequence of words, we consider each of the query word as representing a concept  $c_i$ . In addition to the given word form,  $c_i$  can also be expressed by other alternative forms. However, the appropriate alterations do not only depend on the original word of  $c_i$ , but also on other query words or their alterations.

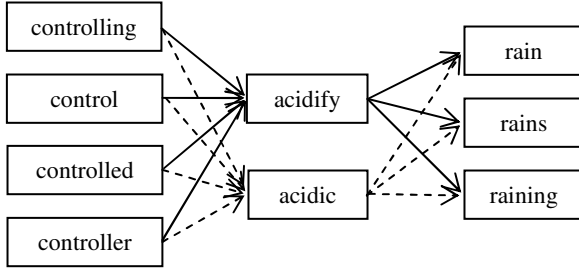


Figure 1: Considering all Combinations to Calculate the Plausibility of Alterations

Accordingly, a confidence weight is determined for each alteration candidate. For example, in the query “Steve Jobs at Apple”, the alteration “job” of “jobs” should have a low confidence; while in the query “finding jobs in Apple”, it should have a high confidence.

One way to measure the confidence of an alteration is the plausibility of its appearing in the query. Since each concept may be expressed by several alterations, we consider all the alterations of context concepts when calculating the plausibility of a given word. Suppose we have the query “controlling acid rain”. The second concept has two alterations - “acidify” and “acidic”. For each of the alterations, our method will consider all the combinations with other words, as illustrated in figure 1, where each combination is shown as a path. More precisely, for a query of  $n$  words (or their corresponding concepts), let  $e_{ij} \in c_i$ ,  $j=1,2,\dots,|c_i|$  be the alterations of concept  $c_i$ . Then we have:

$$P(e_{ij}) = \sum_{1,j_1=1}^{|c_1|} \sum_{2,j_2=1}^{|c_2|} \dots \sum_{i-1,j_{i-1}=1}^{|c_{i-1}|} \sum_{i+1,j_{i+1}=1}^{|c_{i+1}|} \dots \sum_{n,j_n=1}^{|c_n|} P(e_{1,j_1}, e_{2,j_2}, \dots, e_{i,j_i}, \dots, e_{n,j_n}) \quad (1)$$

In equation 1,  $e_{1,j_1}, e_{2,j_2}, \dots, e_{i,j_i}, \dots, e_{n,j_n}$  is a path passing through  $e_{i,j}$ . For simplicity, we abbreviate it as  $e_1 e_2 \dots e_i \dots e_n$ . In this work, we used bigram language model to calculate the probability of each path. Then we have:

$$P(e_1, e_2, \dots, e_i, \dots, e_n) = P(e_1) \prod_{k=2}^n P(e_k | e_{k-1}) \quad (2)$$

$P(e_k | e_{k-1})$  is estimated with a back-off bigram language model (Goodman, 2001). In the experiments with TREC6&7&8, we train the model with all text collections; while in the experiments with Gov2 data, we only used about 1/10 of the GOV2 data to train the bigram model because the whole Gov2 collection is too large.

Directly calculating  $P(e_{ij})$  by summing the probabilities of all paths passing through  $e_{ij}$  is an NP problem (Rabiner, 1989), and is intractable if the query is long. Therefore, we use the forward-backward algorithm (Bishop, 2006) to calculate  $P(e_{ij})$  in a more efficient way. After calculating  $P(e_{ij})$  for each  $c_i$ , we select one alteration which has the highest probability. We limit the number of additional alterations to 1 in order to limit query traffic. Our experiments will show that this is often sufficient.

## 5 Regression Model for Alteration Selection

None of the previous selection methods considers how well an alteration would perform in retrieval. The Bigram Expansion model assumes that the query replaced with better alterations should have a higher likelihood. This approach belongs to the family of unsupervised learning. In this section, we introduce a method belonging to supervised learning family. This method develops a regression model from a set of training data, and it is capable of predicting the expected change in performance when the original query is augmented by this alteration. The performance change is measured by the difference in the Mean Average Precision (MAP) between the augmented and the original query. The training instances are defined by the original query string, an original query term under consideration and one alteration to the query term. A set of features will be used, which will be defined later in this section.

### 5.1 Linear Regression Model

The goal of the regression model is to predict the performance change when a query term is augmented with an alteration. There are several regression models, ranging from the simplest linear regression model to non-linear alternatives, such as a neural network (Duda et al., 2001), a Regression SVM (Bishop, 2006). For simplicity, we use linear regression model here. We denote an instance in the feature space as  $X$ , and the weights of features are denoted as  $W$ . Then the linear regression model is defined as:

$$f(X) = W^T X \quad (3)$$

where  $W^T$  is the transpose of  $W$ . However, we will have a technical problem if we set the target value to the performance change directly: The range of

values of  $f(X)$  is  $(-\infty, +\infty)$ , while the range of performance change is  $[-1, 1]$ . The two value ranges do not match. This inconsistency may result in severe problems when the scales of feature values vary dramatically (Duda et al., 2001). To solve this problem, we do a simple transformation on the performance change. Let the change be  $y \in [-1, 1]$ , then the transformed performance change is:

$$\varphi(y) = \log \frac{1+y+\gamma}{1-y+\gamma} \quad y \in [-1, 1] \quad (4)$$

where  $\gamma$  is a very small positive real number (set to be  $1e-37$  in the experiments), which acts as a smoothing factor. The value of  $\varphi(y)$  can be an arbitrary real number.  $\varphi(y)$  is a monotonic function defined in the range of  $[-1, 1]$ . Moreover, the fixed point of  $\varphi(y)$  is 0, i.e.,  $\varphi(y) = y$  when  $y=0$ . This property is nice; it means that the expansion brings positive improvement if and only if  $f(X) > 0$ , which makes it easy to determine which alteration is better.

We train the regression model by minimizing the mean square error. Suppose there are training instances  $X_1, X_2, \dots, X_m$ , and the corresponding performance change is  $y_i, i=1, 2, \dots, m$ . We calculate the mean square error with the following equation:

$$err(W) = \sum_{i=1}^m (W^T X_i - \varphi(y_i))^2 \quad (5)$$

Then the optimal weight is defined as:

$$W^* = \arg \min_W err(W) \quad (6)$$

$$= \arg \min_W \sum_{i=1}^m (W^T X_i - \varphi(y_i))^2$$

Because  $err(W)$  is a convex function of  $W$ , it has a global minimum and obtains its minimum when the gradient is zero (Bazaraa et al., 2006). Then we have:

$$\frac{\partial err(W^*)}{\partial W^*} = \sum_{i=1}^m (W^T X_i - \varphi(y_i)) X_i^T = 0$$

$$\text{So, } W^{*T} \sum_{i=1}^m X_i X_i^T = \sum_{i=1}^m \varphi(y_i) X_i^T$$

In fact,  $\sum_{i=1}^m X_i X_i^T$  is a square matrix, we denote it as  $XX^T$ . Then we have:

$$W^* = (XX^T)^{-1} \left[ \sum_{i=1}^m \varphi(y_i) X_i^T \right] \quad (7)$$

The matrix  $XX^T$  is an  $l \times l$  square matrix, where  $l$  is the number of features. In our experiments, we only use three features. Therefore the optimal weights can be calculated efficiently even we have a large number of training instances.

## 5.2 Constructing Training Data

As a supervised learning method, the regression model is trained with a set of training data. We illustrate here the procedure to generate training instances with an example.

Given a query “controlling acid rain”, we obtain the MAP of the original query at first. Then we augment the query with an alteration to the original term (one term at a time) at each time. We retain the MAP of the augmented query and compare it with the original query to obtain the performance change. For this query, we expand “controlling” by “control” and get an augmented query “(controlling OR control) acid rain”. We can obtain the difference between the MAP of the augmented query and that of the original query. By doing this, we can generate a series of training instances consisting of the original query string, the original query term under consideration, its alteration and the performance change, for example:

*<controlling acid rain, controlling, control, 0.05>*

Note that we use MAP to measure performance, but we could well use other metrics such as NDCG (Peng et al., 2007) or P@N (precision at top-N documents).

## 5.3 Features Used for Regression Model

Three features are used. The first feature reflects to what degree an alteration is coherent with the other terms. For example, for the query “controlling acid rain”, the coherence of the alteration “acidic” is measured by the logarithm of its co-occurrence with the other query terms within a predefined window (90 words) in the corpus. That is:

$\log(\text{count}(\text{controlling} \dots \text{acidic} \dots \text{rain} | \text{window}) + 0.5)$  where “...” means there may be some words between two query terms. Word order is ignored.

The second feature is an extension to point-wise mutual information (Rijsbergen, 1979), defined as follows:

$$\log \left( \frac{P(\text{controlling} \dots \text{acidic} \dots \text{rain} | \text{window})}{P(\text{controlling})P(\text{acidic})P(\text{rain})} \right)$$

where  $P(\text{controlling} \dots \text{acidic} \dots \text{rain} | \text{window})$  is the co-occurrence probability of the trigram containing acidic within a predefined window (50 words).  $P(\text{controlling})$ ,  $p(\text{acidic})$ ,  $P(\text{rain})$  are probabilities of the three words in the collection. The three words are defined as: the term under consideration, the first term to the left of that term, and the first term to the right. If a query contains less than 3

terms or the term under consideration is the beginning/ending term in the query, we will set the probability of the missed term/terms to be 1. Therefore, it becomes point-wise mutual information when the query contains only two terms. In fact, this feature is supplemental to the first feature. When the query is very long and the first feature always obtains a value of  $\log(0.5)$ , so it does not have any discriminative ability. On the other hand, the second feature helps because it can capture some co-occurrence information no matter how long the query is.

The last feature is the bias, whose value is always set to be 1.0.

The regression model is trained in a leave-one-out cross-validation manner on three collections; each of them is used in turn as a test collection while the two others are used for training. For each incoming query, the regression model predicts the expected performance change when one alteration is used. For each query term, we only select the alteration with the largest positive performance change. If none of its alterations produce a positive performance change, we do not expand the query term. This selection is therefore more restrictive than the Bigram Expansion Model. Nevertheless, our experiments show that it improves retrieval effectiveness further.

## 6 Experiments

### 6.1 Experiment Settings

In this section, our aim is to evaluate the two context-sensitive word alteration selection methods. The ideal evaluation corpus should be composed of some Web data. Unfortunately, such data are not publicly available and the results also could not be compared with other published results. Therefore, we use two TREC collections. The first one is the ad-hoc retrieval test collections used for TREC6&7& 8. This collection is relative small and homogeneous. The second one is the Gov2 data. It is obtained by crawling the entire .gov domain and has been used for three TREC Terabyte tracks (TREC2004-2006). Table 1 shows some statistics of the two collections. For each collection, we use 150 queries. Since the Regression model needs some data for training, we divided the queries into three parts, each containing 50 queries. We then use leave-one-out cross-validation. The evaluation metrics shown below are the average value of the

Name	Description	Size (GB)	#Doc	Query
<b>TREC6 &amp;7&amp;8</b>	TREC disk4&5, Newspapers	1.7	500,447	301-450
<b>Gov2</b>	2004 crawl of entire .gov domain	427	25,205,179	701-850

Table1: Overview of Test Collections

three-fold cross-validation. Because the queries in Web are usually very short, we use only the title field of each query.

To correspond to Web search practice, both documents and queries are not stemmed. We do not filter the stop words either.

Two main metrics are used: the Mean Average Precision (MAP) for the top 1000 documents to measure retrieval effectiveness, and the number of terms in the query to reflect query traffic. In addition, we also provide precision for the top 30 documents (P@30) to show the impact on top ranked documents. We also conducted t-tests to determine whether the improvement is statistically significant.

The Indri 2.5 search engine (Strohman et al., 2004) is used as our basic retrieval system. It provides for a rich query language allowing disjunctive combinations of words in queries.

### 6.2 Experimental Results

The first baseline method we compare with only uses the original query, which is named *Original*. In addition to this, we also compare with the following methods:

*Naïve Exp*: The Naïve expansion model expands each query term with all terms in the vocabulary sharing the same root with it. This model is equivalent to the traditional stemming method.

*UMASS*: This is the result reported in (Metzler et al., 2006) using Porter stemming for both document and query terms. This reflects a state-of-the-art result using Porter stemming.

*Similarity*: We select the alterations (at most 5) with the highest similarity to the original term. This is the method described in section 3.

The two methods we propose in this paper are the following ones:

*Bigram Exp*: the alteration is chosen by a Bigram Expansion model.

*Regression*: the alteration is chosen by a Regression model.

Model	P@30	#term	MAP	Imp.
<b>Original</b>	0.4701	158	0.2440	----
<b>UMASS</b>	-----	-----	0.2666	9.26
<b>Naïve Exp</b>	0.4714	1345	0.2653	8.73
<b>Similarity</b>	0.4900	303	0.2689	10.20*
<b>Bigram Exp</b>	0.5007	303	0.2751	12.75**
<b>Regression</b>	0.5054	237	0.2773	13.65**

Table 2: Results of Query 701-750 Over Gov2 Data

Model	P@30	#term	MAP	Imp.
<b>Original</b>	0.4907	158	0.2738	----
<b>UMASS</b>	-----	-----	0.3251	18.73
<b>Naïve Exp</b>	0.5213	1167	0.3224	17.75**
<b>Similarity</b>	0.5140	290	0.3043	11.14**
<b>Bigram Exp.</b>	0.5153	290	0.3107	13.47**
<b>Regression</b>	0.5140	256	0.3144	14.82**

Table 3: Results of Query 751-800 over Gov2 Data

Model	P@30	#term	MAP	Imp.
<b>Original</b>	0.4710	154	0.2887	----
<b>UMASS</b>	-----	-----	0.2996	3.78
<b>Naïve Exp</b>	0.4633	1225	0.2999	3.87
<b>Similarity</b>	0.4710	288	0.2976	3.08
<b>Bigram Exp</b>	0.4730	288	0.3137	8.66**
<b>Regression</b>	0.4748	237	0.3118	8.00*

Table 4: Results of Query 801-850 over Gov2 Data

Model	P@30	#term	MAP	Imp.
<b>Original</b>	0.2673	137	0.1669	----
<b>Naïve Exp</b>	0.3053	783	0.2146	28.57**
<b>Similarity</b>	0.3007	255	0.2020	21.03**
<b>Bigram Exp</b>	0.3033	255	0.2091	25.28**
<b>Regression</b>	0.3113	224	0.2161	29.48**

Table 5: Results of Query 301-350 over TREC6&7&8

Model	P@30	#term	MAP	Imp.
<b>Original</b>	0.2820	126	0.1639	----
<b>Naïve Exp</b>	0.2787	736	0.1665	1.59
<b>Similarity</b>	0.2867	244	0.1650	0.67
<b>Bigram Exp.</b>	0.2800	244	0.1641	0.12
<b>Regression</b>	0.2867	214	0.1664	1.53

Table 6: Results of Query 351-400 over TREC6&7&8

Model	P@30	#term	MAP	Imp.
<b>Original</b>	0.2833	124	0.1759	----
<b>Naïve Exp</b>	0.3167	685	0.2138	21.55**
<b>Similarity</b>	0.3080	240	0.2066	17.45**
<b>Bigram Exp</b>	0.3133	240	0.2080	18.25**
<b>Regression</b>	0.3220	187	0.2144	21.88**

Table7: Results of Query 401-450 over TREC6&7&8

Tables 2, 3, 4 show the results of Gov2 data while table 5, 6, 7 show the results of the TREC6&7&8 collection. In the tables, the \* mark indicates that the improvement over the original

model is statistically significant with  $p\text{-value} < 0.05$ , and \*\* means the  $p\text{-values} < 0.01$ .

From the tables, we see that both word stemming (UMASS) and expansion with word alterations can improve MAP for all six tasks. In most cases (except in table 4 and 6), it also improve the precision of top ranked documents. This shows the usefulness of word stemming or word alteration expansion for IR.

We can make several additional observations:

- 1). Stemming Vs Expansion. UMASS uses document and query stemming while *Naive Exp* uses expansion by word alteration. We stated that both approaches are equivalent. The equivalence is confirmed by our experiment results: for all Gov2 collections, these approaches perform equivalently.
- 2). The Similarity model performs very well. Compared with the Naïve Expansion model, it produces quite similar retrieval effectiveness, while the query traffic is dramatically reduced. This approach is similar to the work of Xu and Croft (1998), and can be considered as another state-of-the-art result.
- 3). In comparison, the Bigram Expansion model performs better than the Similarity model. This shows that it is useful to consider query context in selecting word alterations.
- 4). The Regression model performs the best of all the models. Compared with the Original query, it adds fewer than 2 alterations for each query on average (since each group has 50 queries); nevertheless we obtained improvements on all the six collections. Moreover, the improvements on five collections are statistically significant. It also performs slightly better than the Similarity and Bigram Expansion methods, but with fewer alterations. This shows that the supervised learning approach, if used in the correct way, is superior to an unsupervised approach. Another advantage over the two other models is that the Regression model can reduce the number of alterations further. Because the Regression model selects alterations according to their expected improvement, the improvement of the alterations to one query term can be compared with that of the alterations to other query terms. Therefore, we can select at most one optimal alteration for the whole query. However, with the Similarity or Bigram Expansion models, the selection value, either similarity or query likelihood, cannot be

compared across the query terms. As a consequence, more alterations need to be selected, leading to heavier query traffic.

## 7 Conclusion

Traditional IR approaches stem terms in both documents and queries. This approach is appropriate for general purpose IR, but is ill-suited for the specific retrieval needs in Web search such as quoted queries or queries with a specific word form that should not be stemmed. The current practice in Web search is not to stem words in index, but rather to perform a form of expansion using word alteration.

However, a naïve expansion will result in many alterations and this will increase the query traffic. This paper has proposed two alternative methods to select precise alterations by considering the query context. We seek to produce similar or better improvements in retrieval effectiveness, while limiting the query traffic.

In the first method proposed – the Bigram Expansion model, query context is modeled by a bigram language model. For each query term, the selected alteration is the one which maximizes the query likelihood. In the second method - Regression model, we fit a regression model to calculate the expected improvement when the original query is expanded by an alteration. Only the alteration that is expected to yield the largest improvement to retrieval effectiveness is added.

The proposed methods were evaluated on two TREC benchmarks: the ad-hoc retrieval test collection for TREC6&7&8 and the Gov2 data. Our experimental results show that both proposed methods perform significantly better than the original queries. Compared with traditional word stemming or the naïve expansion approach, our methods can not only improve retrieval effectiveness, but also greatly reduce the query traffic.

This work shows that query expansion with word alterations is a reasonable alternative to word stemming. It is possible to limit the query traffic by a query-dependent selection of word alterations. Our work shows that both unsupervised and supervised learning can be used to perform alteration selection.

Our methods can be further improved in several aspects. For example, we could integrate other features in the regression model, and use other non-linear regression models, such as Bayesian regres-

sion models (e.g. Gaussian Process regression) (Rasmussen and Williams, 2006). The additional advantage of these models is that we can not only obtain the expected improvement in retrieval effectiveness for an alteration, but also the probability of obtaining an improvement (i.e. the robustness of the alteration).

Finally, it would be interesting to test the approaches using real Web data.

## References

- Anick, P. (2003) Using Terminological Feedback for Web Search Refinement: a Log-based Study. In SIGIR, pp. 88-95.
- Bazaraa, M., Sherali, H., and Shett, C. (2006). Nonlinear Programming, Theory and Algorithms. John Wiley & Sons Inc.
- Bishop, C. (2006). Pattern Recognition and Machine Learning. Springer.
- Duda, R., Hart, P., and Stork, D. (2001). Pattern Classification, John Wiley & Sons, Inc.
- Goodman, J. (2001). A Bit of Progress in Language Modeling. Technical report.
- Jones, R., Rey, B., Madani, O., and Greiner, W. (2006). Generating Query Substitutions. In WWW2006, pp. 387-396
- Kraaij, W. and Pohlmann, R. (1996) Viewing Stemming as Recall Enhancement. Proc. SIGIR, pp. 40-48.
- Krovetz, R. (1993). Viewing Morphology as an Inference Process. Proc. ACM SIGIR, pp. 191-202.
- Lin, D. (1998). Automatic Retrieval and Clustering of Similar Words. In COLING-ACL, pp. 768-774.
- Metzler, D., Strohman, T. and Croft, B. (2006). Indri TREC Notebook 2006: Lessons learned from Three Terabyte Tracks. In the Proceedings of TREC 2006.
- Peng, F., Ahmed, N., Li, X., and Lu, Y. (2007). Context Sensitive Stemming for Web Search. Proc. ACM SIGIR, pp. 639-636 .
- Porter, M. (1980) An Algorithm for Suffix Stripping. Program, 14(3): 130-137.
- Rabiner, L. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In Proceedings of IEEE Vol. 77(2), pp. 257-286.
- Rijsbergen, V. (1979). Information Retrieval. Butterworths, second version.
- Strohman, T., Metzler, D. and Turtle, H., and Croft, B. (2004). Indri: A Language Model-based Search Engine for Complex Queries. In Proceedings of the International conference on Intelligence Analysis.
- Xu, J. and Croft, B. (1996). Query Expansion Using Local and Global Document Analysis. Proc. ACM SIGIR, pp. 4-11.
- Xu, J. and Croft, B. (1998). Corpus-based Stemming Using Co-occurrence of Word Variants. ACM TOIS, 16(1): 61-81.

# Searching Questions by Identifying Question Topic and Question Focus

Huizhong Duan<sup>1</sup>, Yunbo Cao<sup>1,2</sup>, Chin-Yew Lin<sup>2</sup> and Yong Yu<sup>1</sup>

<sup>1</sup>Shanghai Jiao Tong University,  
Shanghai, China, 200240  
{summer, yyu}@apex.sjtu.edu.cn

<sup>2</sup>Microsoft Research Asia,  
Beijing, China, 100080  
{yunbo.cao, cyl}@microsoft.com

## Abstract

This paper is concerned with the problem of question search. In question search, given a question as query, we are to return questions semantically equivalent or close to the queried question. In this paper, we propose to conduct question search by identifying question topic and question focus. More specifically, we first summarize questions in a data structure consisting of question topic and question focus. Then we model question topic and question focus in a language modeling framework for search. We also propose to use the MDL-based tree cut model for identifying question topic and question focus automatically. Experimental results indicate that our approach of identifying question topic and question focus for search significantly outperforms the baseline methods such as Vector Space Model (VSM) and Language Model for Information Retrieval (LMIR).

## 1 Introduction

Over the past few years, online services have been building up very large archives of questions and their answers, for example, traditional FAQ services and emerging community-based Q&A services (e.g., Yahoo! Answers<sup>1</sup>, Live QnA<sup>2</sup>, and Baidu Zhidao<sup>3</sup>).

To make use of the large archives of questions and their answers, it is critical to have functionality facilitating users to search previous answers. Typically, such functionality is achieved by first retrieving questions expected to have the same answers as a queried question and then returning the related answers to users. For example, given question *Q1* in Table 1, question *Q2* can be re-

turned and its answer will then be used to answer *Q1* because the answer of *Q2* is expected to partially satisfy the queried question *Q1*. This is what we called *question search*. In question search, returned questions are *semantically equivalent or close* to the queried question.

<b>Query:</b> <i>Q1: Any cool clubs in Berlin or Hamburg?</i>
<b>Expected:</b> <i>Q2: What are the best/most fun clubs in Berlin?</i>
<b>Not Expected:</b> <i>Q3: Any nice hotels in Berlin or Hamburg?</i> <i>Q4: How long does it take to Hamburg from Berlin?</i> <i>Q5: Cheap hotels in Berlin?</i>

Table 1. An Example on Question Search

Many methods have been investigated for tackling the problem of question search. For example, Jeon et al. have compared the uses of four different retrieval methods, i.e. vector space model, Okapi, language model, and translation-based model, within the setting of question search (Jeon et al., 2005b). However, all the existing methods treat questions just as plain texts (without considering question structure). For example, obviously, *Q2* can be considered semantically closer to *Q1* than *Q3-Q5* although all questions (*Q2-Q5*) are related to *Q1*. The existing methods are not able to tell the difference between question *Q2* and questions *Q3*, *Q4*, and *Q5* in terms of their relevance to question *Q1*. We will clarify this in the following.

In this paper, we propose to conduct question search by identifying question topic and question focus.

The question topic usually represents the major context/constraint of a question (e.g., Berlin, Hamburg) which characterizes users' interests. In contrast, question focus (e.g., cool club, cheap hotel) presents certain aspect (or descriptive features) of the question topic. For the aim of retrieving semantically equivalent (or close) questions, we need to

<sup>1</sup> <http://answers.yahoo.com>

<sup>2</sup> <http://qna.live.com>

<sup>3</sup> <http://zhidao.baidu.com>



assure that returned questions are related to the queried question with respect to both question topic and question focus. For example, in Table 1,  $Q2$  preserves certain useful information of  $Q1$  in the aspects of both question topic (Berlin) and question focus (fun club) although it loses some useful information in question topic (Hamburg). In contrast, questions  $Q3$ - $Q5$  are not related to  $Q1$  in question focus (although being related in question topic, e.g. Hamburg, Berlin), which makes them unsuitable as the results of question search.

We also propose to use the MDL-based (Minimum Description Length) tree cut model for automatically identifying question topic and question focus. Given a question as query, a structure called *question tree* is constructed over the question collection including the queried question and all the related questions, and then the MDL principle is applied to find a *cut* of the question tree specifying the question topic and the question focus of each question.

In a summary, we summarize questions in a data structure consisting of *question topic* and *question focus*. On the basis of this, we then propose to model question topic and question focus in a language modeling framework for search. To the best of our knowledge, none of the existing studies addressed question search by modeling both question topic and question focus.

We empirically conduct the question search with questions about ‘travel’ and ‘computers & internet’. Both kinds of questions are from Yahoo! Answers. Experimental results show that our approach can significantly improve traditional methods (e.g. VSM, LMIR) in retrieving relevant questions.

The rest of the paper is organized as follow. In Section 2, we present our approach to question search which is based on identifying question topic and question focus. In Section 3, we empirically verify the effectiveness of our approach to question search. In Section 4, we employ a translation-based retrieval framework for extending our approach to fix the issue called ‘lexical chasm’. Section 5 surveys the related work. Section 6 concludes the paper by summarizing our work and discussing the future directions.

## 2 Our Approach to Question Search

Our approach to question search consists of two steps: (a) summarize questions in a data structure consisting of question topic and question focus; (b)

model question topic and question focus in a language modeling framework for search.

In the step (a), we employ the MDL-based (Minimum Description Length) tree cut model for automatically identifying question topic and question focus. Thus, this section will begin with a brief review of the MDL-based tree cut model and then follow that by an explanation of steps (a) and (b).

### 2.1 The MDL-based tree cut model

Formally, a tree cut model  $M$  (Li and Abe, 1998) can be represented by a pair consisting of a tree cut  $\Gamma$ , and a probability parameter vector  $\theta$  of the same length, that is,

$$M = (\Gamma, \theta) \quad (1)$$

where  $\Gamma$  and  $\theta$  are

$$\Gamma = [C_1, C_2, \dots, C_k], \quad (2)$$

$$\theta = [p(C_1), p(C_2), \dots, p(C_k)]$$

where  $C_1, C_2, \dots, C_k$  are classes determined by a cut in the tree and  $\sum_{i=1}^k p(C_i) = 1$ . A ‘cut’ in a tree is any set of nodes in the tree that defines a partition of all the nodes, viewing each node as representing the set of child nodes as well as itself. For example, the cut indicated by the dash line in Figure 1 corresponds to three classes:  $[n_0, n_{11}], [n_{13}, n_{24}]$ , and  $[n_{12}, n_{21}, n_{22}, n_{23}]$ .

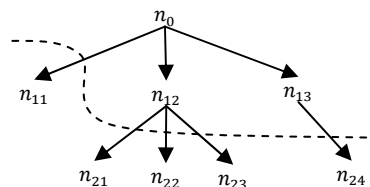


Figure 1. An Example on the Tree Cut Model

A straightforward way for determining a cut of a tree is to collapse the nodes of less frequency into their parent nodes. However, the method is too heuristic for it relies much on manually tuned frequency threshold. In our practice, we turn to use a theoretically well-motivated method based on the MDL principle. MDL is a principle of data compression and statistical estimation from information theory (Rissanen, 1978).

Given a sample  $S$  and a tree cut  $\Gamma$ , we employ MLE to estimate the parameters of the corresponding tree cut model  $\hat{M} = (\Gamma, \hat{\theta})$ , where  $\hat{\theta}$  denotes the estimated parameters.

According to the MDL principle, the description length (Li and Abe, 1998)  $L(\hat{M}, S)$  of the tree cut model  $\hat{M}$  and the sample  $S$  is the sum of the model

description length  $L(\Gamma)$ , the parameter description length  $L(\hat{\theta}|\Gamma)$ , and the data description length  $L(S|\Gamma, \hat{\theta})$ , i.e.

$$L(\hat{M}, S) = L(\Gamma) + L(\hat{\theta}|\Gamma) + L(S|\Gamma, \hat{\theta}) \quad (3)$$

The model description length  $L(\Gamma)$  is a subjective quantity which depends on the coding scheme employed. Here, we simply assume that each tree cut model is equally likely *a priori*.

The parameter description length  $L(\hat{\theta}|\Gamma)$  is calculated as

$$L(\hat{\theta}|\Gamma) = \frac{k}{2} \times \log |S| \quad (4)$$

where  $|S|$  denotes the sample size and  $k$  denotes the number of free parameters in the tree cut model, i.e.  $k$  equals the number of nodes in  $\Gamma$  minus one.

The data description length  $L(S|\Gamma, \hat{\theta})$  is calculated as

$$L(S|\Gamma, \hat{\theta}) = -\sum_{n \in S} \log \hat{p}(n) \quad (5)$$

where

$$\hat{p}(n) = \frac{1}{|C|} \times \frac{f(C)}{|S|} \quad (6)$$

where  $C$  is the class that  $n$  belongs to and  $f(C)$  denotes the total frequency of instances in class  $C$  in the sample  $S$ .

With the description length defined as (3), we wish to select a tree cut model with the minimum description length and output it as the result. Note that the model description length  $L(\Gamma)$  can be ignored because it is the same for all tree cut models.

The MDL-based tree cut model was originally introduced for handling the problem of generalizing case frames using a thesaurus (Li and Abe, 1998). To the best of our knowledge, no existing work utilizes it for question search. This may be partially because of the unavailability of the resources (e.g., thesaurus) which can be used for embodying the questions in a tree structure. In Section 2.2, we will introduce a tree structure called *question tree* for representing questions.

## 2.2 Identifying question topic and question focus

In principle, it is possible to identify *question topic* and *question focus* of a question by only parsing the question itself (for example, utilizing a syntactic parser). However, such a method requires accurate parsing results which cannot be obtained from the noisy data from online services.

Instead, we propose using the MDL-based tree cut model which identifies question topics and

question foci for a set of questions together. More specifically, the method consists of two phases:

- 1) Constructing a *question tree*: represent the queried question and all the related questions in a tree structure called *question tree*;
- 2) Determining a *tree cut*: apply the MDL principle to the *question tree*, which yields the cut specifying *question topic* and *question focus*.

### 2.2.1 Constructing a question tree

In the following, with a series of definitions, we will describe how a *question tree* is constructed from a collection of questions.

Let's begin with explaining the representation of a question. A straightforward method is to represent a question as a bag-of-words (possibly ignoring stop words). However, this method cannot discern 'the hotels in Paris' from 'the Paris hotel'. Thus, we turn to use the linguistic units carrying on more semantic information. Specifically, we make use of two kinds of units: BaseNP (Base Noun Phrase) and WH-ngram. A BaseNP is defined as a simple and non-recursive noun phrase (Cao and Li, 2002). A WH-ngram is an ngram beginning with WH-words. The WH-words that we consider include 'when', 'what', 'where', 'which', and 'how'. We refer to these two kinds of units as '*topic terms*'. With 'topic terms', we represent a question as a *topic chain* and a set of questions as a *question tree*.

**Definition 1 (Topic Profile)** The *topic profile*  $\theta_t$  of a topic term  $t$  in a categorized question collection is a probability distribution of categories  $\{p(c|t)\}_{c \in C}$  where  $C$  is a set of categories.

$$p(c|t) = \frac{\text{count}(c,t)}{\sum_{c \in C} \text{count}(c,t)} \quad (7)$$

where  $\text{count}(c,t)$  is the frequency of the topic term  $t$  within category  $c$ . Clearly, we have  $\sum_{c \in C} p(c|t) = 1$ .

By 'categorized questions', we refer to the questions that are organized in a tree of taxonomy. For example, at Yahoo! Answers, the question "How do I install my wireless router" is categorized as "Computers & Internet  $\rightarrow$  Computer Networking". Actually, we can find categorized questions at other online services such as FAQ sites, too.

**Definition 2 (Specificity)** The *specificity*  $s(t)$  of a topic term  $t$  is the inverse of the entropy of the topic profile  $\theta_t$ . More specifically,

$$s(t) = 1 / (-\sum_{c \in C} p(c|t) \log p(c|t) + \varepsilon) \quad (8)$$

where  $\varepsilon$  is a smoothing parameter used to cope with the topic terms whose entropy is 0. In our experiments, the value of  $\varepsilon$  was set 0.001.

We use the term *specificity* to denote how specific a topic term is in characterizing information needs of users who post questions. A topic term of high specificity (e.g., Hamburg, Berlin) usually specifies the *question topic* corresponding to the main context of a question because it tends to occur only in a few categories. A topic term of low specificity is usually used to represent the *question focus* (e.g., cool club, where to see) which is relatively volatile and might occur in many categories.

**Definition 3 (Topic Chain)** A topic chain  $q^c$  of a question  $q$  is a sequence of ordered topic terms  $t_1 \rightarrow t_2 \rightarrow \dots \rightarrow t_m$  such that

- 1)  $t_i$  is included in  $q$ ,  $1 \leq i \leq m$ ;
- 2)  $s(t_k) > s(t_l)$ ,  $1 \leq k < l \leq m$ .

For example, the topic chain of “any cool clubs in Berlin or Hamburg?” is “Hamburg  $\rightarrow$  Berlin  $\rightarrow$  cool club” because the *specificities* for ‘Hamburg’, ‘Berlin’, and ‘cool club’ are 0.99, 0.62, and 0.36.

**Definition 4 (Question Tree)** A question tree of a question set  $Q = \{q_i\}_{i=1}^N$  is a prefix tree built over the topic chains  $Q^c = \{q_i^c\}_{i=1}^N$  of the question set  $Q$ . Clearly, if a question set contains only one question, its question tree will be exactly same as the topic chain of the question.

Note that the root node of a question tree is associated with *empty string* as the definition of prefix tree requires (Fredkin, 1960).

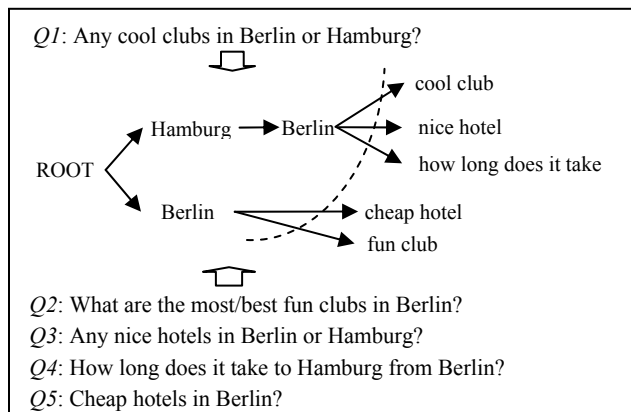


Figure 2. An Example of a Question Tree

Given the topic chains with respect to the questions in Table 1 as follow,

- $Q1$ : Hamburg  $\rightarrow$  Berlin  $\rightarrow$  cool club
- $Q2$ : Berlin  $\rightarrow$  fun club
- $Q3$ : Hamburg  $\rightarrow$  Berlin  $\rightarrow$  nice hotel

- $Q4$ : Hamburg  $\rightarrow$  Berlin  $\rightarrow$  how long does it take
- $Q5$ : Berlin  $\rightarrow$  cheap hotel

we can have the question tree presented in Figure 2.

### 2.2.2 Determining the tree cut

According to the definition of a *topic chain*, the topic terms in a topic chain of a question are ordered by their specificity values. Thus, a cut of a topic chain naturally separates the topic terms of low specificity (representing question focus) from the topic terms of high specificity (representing question topic). Given a topic chain of a question consisting of  $m$  topic terms, there exist  $(m - 1)$  possible cuts. The question is: which cut is the best?

We propose using the MDL-based tree cut model for the search of the best cut in a topic chain. Instead of dealing with each topic chain individually, the proposed method handles a set of questions together. Specifically, given a queried question, we construct a question tree consisting of both the queried question and the related questions, and then apply the MDL principle to select the best cut of the question tree. For example, in Figure 2, we hope to get the cut indicated by the dashed line. The topic terms on the left of the dashed line represent the question topic and those on the right of the dashed line represent the question focus. Note that the tree cut yields a cut for each individual topic chain (each path) within the question tree accordingly.

A cut of a topic chain  $q^c$  of a question  $q$  separates the topic chain in two parts: HEAD and TAIL. HEAD (denoted as  $H(q^c)$ ) is the subsequence of the original topic chain  $q^c$  before the cut. TAIL (denoted as  $T(q^c)$ ) is the subsequence of  $q^c$  after the cut. Thus,  $q^c = H(q^c) \rightarrow T(q^c)$ . For instance, given the tree cut specified in Figure 2, for the topic chain of  $Q1$  “Hamburg  $\rightarrow$  Berlin  $\rightarrow$  cool club”, the HEAD and TAIL are “Hamburg  $\rightarrow$  Berlin” and “cool club” respectively.

### 2.3 Modeling question topic and question focus for search

We employ the framework of language modeling (for information retrieval) to develop our approach to question search.

In the language modeling approach to information retrieval, the relevance of a targeted question  $\tilde{q}$  to a queried question  $q$  is given by the probability  $p(q|\tilde{q})$  of generating the queried question  $q$

from the language model formed by the targeted question  $\tilde{q}$ . The targeted question  $\tilde{q}$  is from a collection  $C$  of questions.

Following the framework, we propose a mixture model for modeling question structure (namely, question topic and question focus) within the process of searching questions:

$$p(q|\tilde{q}) = \lambda \cdot p(H(q)|H(\tilde{q})) + (1 - \lambda) \cdot p(T(q)|T(\tilde{q})) \quad (9)$$

In the mixture model, it is assumed that the process of generating question topics and the process of generating question foci are independent from each other.

In traditional language modeling, a single multinomial model  $p(t|\tilde{q})$  over terms is estimated for each targeted question  $\tilde{q}$ . In our case, two multinomial models  $p(t|H(\tilde{q}))$  and  $p(t|T(\tilde{q}))$  need to be estimated for each targeted question  $\tilde{q}$ .

If unigram document language models are used, the equation (9) can then be re-written as,

$$p(q|\tilde{q}) = \lambda \cdot \prod_{t \in H(q)} p(t|H(\tilde{q}))^{count(q,t)} + (1 - \lambda) \cdot \prod_{t \in T(q)} p(t|T(\tilde{q}))^{count(q,t)} \quad (10)$$

where  $count(q, t)$  is the frequency of  $t$  within  $q$ .

To avoid zero probabilities and estimate more accurate language models, the HEAD and TAIL of questions are smoothed using background collection,

$$p(t|H(\tilde{q})) = \alpha \cdot \hat{p}(t|H(\tilde{q})) + (1 - \alpha) \cdot \hat{p}(t|C) \quad (11)$$

$$p(t|T(\tilde{q})) = \beta \cdot \hat{p}(t|T(\tilde{q})) + (1 - \beta) \cdot \hat{p}(t|C) \quad (12)$$

where  $\hat{p}(t|H(\tilde{q}))$ ,  $\hat{p}(t|T(\tilde{q}))$ , and  $\hat{p}(t|C)$  are the MLE estimators with respect to the HEAD of  $\tilde{q}$ , the TAIL of  $\tilde{q}$ , and the collection  $C$ .

### 3 Experimental Results

We have conducted experiments to verify the effectiveness of our approach to question search. Particularly, we have investigated the use of identifying question topic and question focus for search.

#### 3.1 Dataset and evaluation measures

We made use of the questions obtained from Yahoo! Answers for the evaluation. More specifically, we utilized the *resolved* questions under two of the top-level categories at Yahoo! Answers, namely ‘travel’ and ‘computers & internet’. The questions include 314,616 items from the ‘travel’ category

and 210,785 items from the ‘computers & internet’ category. Each resolved question consists of three fields: ‘title’, ‘description’, and ‘answers’. For search we use only the ‘title’ field. It is assumed that the titles of the questions already provide enough semantic information for understanding users’ information needs.

We developed two test sets, one for the category ‘travel’ denoted as ‘TRL-TST’, and the other for ‘computers & internet’ denoted as ‘CI-TST’. In order to create the test sets, we randomly selected 200 questions for each category.

To obtain the ground-truth of question search, we employed the Vector Space Model (VSM) (Salton et al., 1975) to retrieve the top 20 results and obtained manual judgments. The top 20 results don’t include the queried question itself. Given a returned result by VSM, an assessor is asked to label it with ‘*relevant*’ or ‘*irrelevant*’. If a returned result is considered semantically equivalent (or close) to the queried question, the assessor will label it as ‘*relevant*’; otherwise, the assessor will label it as ‘*irrelevant*’. Two assessors were involved in the manual judgments. Each of them was asked to label 100 questions from ‘TRL-TST’ and 100 from ‘CI-TST’. In the process of manually judging questions, the assessors were presented only the *titles* of the questions (for both the queried questions and the returned questions). Table 2 provides the statistics on the final test set.

	# Queries	# Returned	# Relevant
TRL-TST	200	4,000	256
CI-TST	200	4,000	510

Table 2. Statistics on the Test Data

We utilized two baseline methods for demonstrating the effectiveness of our approach, the VSM and the LMIR (language modeling method for information retrieval) (Ponte and Croft, 1998).

We made use of three measures for evaluating the results of question search methods. They are MAP, R-precision, and MRR.

#### 3.2 Searching questions about ‘travel’

In the experiments, we made use of the questions about ‘travel’ to test the performance of our approach to question search. More specifically, we used the 200 queries in the test set ‘TRL-TST’ to search for ‘relevant’ questions from the 314,616

questions categorized as ‘travel’. Note that only the questions occurring in the test set can be evaluated.

We made use of the taxonomy of questions provided at Yahoo! Answers for the calculation of *specificity of topic terms*. The taxonomy is organized in a tree structure. In the following experiments, we only utilized as the categories of questions the leaf nodes of the taxonomy tree (regarding ‘travel’), which includes 355 categories.

We randomly divided the test queries into five even subsets and conducted 5-fold cross-validation experiments. In each trial, we tuned the parameters  $\lambda$ ,  $\alpha$ , and  $\beta$  in the equation (10)-(12) with four of the five subsets and then applied it to one remaining subset. The experimental results reported below are those averaged over the five trials.

Methods	MAP	R-Precision	MRR
VSM	0.198	0.138	0.228
LMIR	0.203	0.154	0.248
LMIR-CUT	<b>0.236</b>	<b>0.192</b>	<b>0.279</b>

Table 3. Searching Questions about ‘Travel’

In Table 3, our approach denoted by LMIR-CUT is implemented exactly as equation (10). Neither VSM nor LMIR uses the data structure composed of question topic and question focus.

From Table 3, we see that our approach outperforms the baseline approaches VSM and LMIR in terms of all the measures. We conducted a significance test (t-test) on the improvements of our approach over VSM and LMIR. The result indicates that the improvements are statistically significant ( $p$ -value  $< 0.05$ ) in terms of all the evaluation measures.

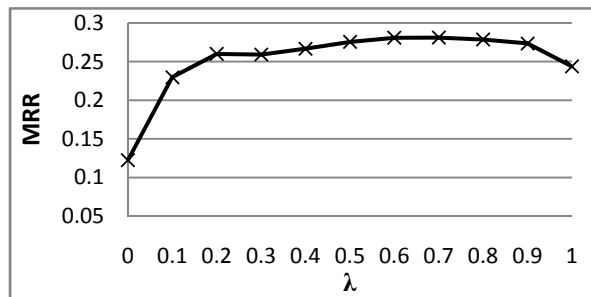


Figure 3. Balancing between Question Topic and Question Focus

In equation (9), we use the parameter  $\lambda$  to balance the contribution of question topic and the contribution of question focus. Figure 3 illustrates how

influential the value of  $\lambda$  is on the performance of question search in terms of MRR. The result was obtained with the 200 queries directly, instead of 5-fold cross-validation. From Figure 3, we see that our approach performs best when  $\lambda$  is around 0.7. That is, our approach tends to emphasize question topic more than question focus.

We also examined the correctness of question topics and question foci of the 200 queried questions. The question topics and question foci were obtained with the MDL-based tree cut model automatically. In the result, 69 questions have incorrect question topics or question foci. Further analysis shows that the errors came from two categories: (a) 59 questions have only the HEAD parts (that is, none of the topic terms fall within the TAIL part), and (b) 10 have incorrect orders of topic terms because the specificities of topic terms were estimated inaccurately. For questions only having the HEAD parts, our approach (equation (9)) reduces to traditional language modeling approach. Thus, even when the errors of category (a) occur, our approach can still work not worse than the traditional language modeling approach. This also explains why our approach performs best when  $\lambda$  is around 0.7. The error category (a) pushes our model to emphasize more in question topic.

Methods	Results
VSM	<ol style="list-style-type: none"> <li>How cold does it usually get in Charlotte, NC during winters?</li> <li>How long and cold are the winters in Rochester, NY?</li> <li><b>How cold is it in Alaska?</b></li> </ol>
LMIR	<ol style="list-style-type: none"> <li><b>How cold is it in Alaska?</b></li> <li>How cold does it get really in Toronto in the winter?</li> <li>How cold does the Mojave Desert get in the winter?</li> </ol>
LMIR-CUT	<ol style="list-style-type: none"> <li><b>How cold is it in Alaska?</b></li> <li><b>How cold is Alaska in March and outdoor activities?</b></li> <li>How cold does it get in Nova Scotia in the winter?</li> </ol>

Table 4. Search Results for ‘How cold does it get in winters in Alaska?’

Table 4 provides the TOP-3 search results which are given by VSM, LMIR, and LMIR-CUT (our approach) respectively. The questions in bold are labeled as ‘relevant’ in the evaluation set. The queried question seeks for the ‘weather’ information about ‘Alaska’. Both VSM and LMIR rank certain

‘irrelevant’ questions higher than ‘relevant’ questions. The ‘irrelevant’ questions are not about ‘Alaska’ although they are about ‘weather’. The reason is that neither VSM nor PVSM is aware that the query consists of the two aspects ‘weather’ (how cold, winter) and ‘Alaska’. In contrast, our approach assures that both aspects are matched. Note that the HEAD part of the topic chain of the queried question given by our approach is “Alaska” and the TAIL part is “winter → how cold”.

### 3.3 Searching questions about ‘computers & internet’

In the experiments, we made use of the questions about ‘computers & internet’ to test the performance of our proposed approach to question search. More specifically, we used the 200 queries in the test set ‘CI-TST’ to search for ‘relevant’ questions from the 210,785 questions categorized as ‘computers & internet’. For the calculation of *specificity of topic terms*, we utilized as the categories of questions the leaf nodes of the taxonomy tree regarding ‘computers & Internet’, which include 23 categories.

We conducted 5-fold cross-validation for the parameter tuning. The experimental results reported in Table 5 are averaged over the five trials.

Methods	MAP	R-Precision	MRR
VSM	0.236	0.175	0.289
LMIR	0.248	0.191	0.304
LMIR-CUT	<b>0.279</b>	<b>0.230</b>	<b>0.341</b>

Table 5. Searching Questions about ‘Computers & Internet’

Again, we see that our approach outperforms the baseline approaches VSM and LMIR in terms of all the measures. We conducted a significance test (t-test) on the improvements of our approach over VSM and LMIR. The result indicates that the improvements are statistically significant (p-value < 0.05) in terms of all the evaluation measures.

We also conducted the experiment similar to that in Figure 3. Figure 4 provides the result. The trend is consistent with that in Figure 3.

We examined the correctness of (automatically identified) question topics and question foci of the 200 queried questions, too. In the result, 65 questions have incorrect question topics or question foci. Among them, 47 fall in the error category (a) and 18 in the error category (b). The distribution of

errors is also similar to that in Section 3.2, which also justifies the trend presented in Figure 4.

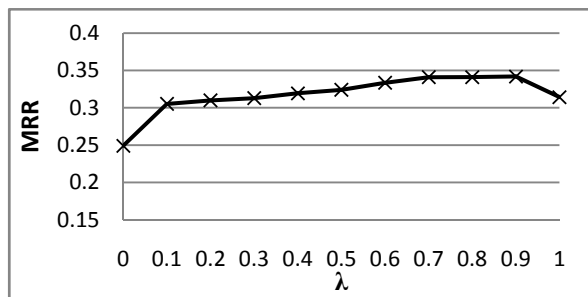


Figure 4. Balancing between Question Topic and Question Focus

## 4 Using Translation Probability

In the setting of question search, besides the topic what we address in the previous sections, another research topic is to fix lexical chasm between questions.

Sometimes, two questions that have the same meaning use very different wording. For example, the questions “where to stay in Hamburg?” and “the best hotel in Hamburg?” have almost the same meaning but are lexically different in question focus (where to stay vs. best hotel). This is the so-called ‘lexical chasm’.

Jeon and Bruce (2007) proposed a mixture model for fixing the lexical chasm between questions. The model is a combination of the language modeling approach (for information retrieval) and translation-based approach (for information retrieval). Our idea of modeling question structure for search can naturally extend to Jeon et al.’s model. More specifically, by using translation probabilities, we can rewrite equation (11) and (12) as follow:

$$p(t|H(\tilde{q})) = \alpha_1 \cdot \hat{p}(t|H(\tilde{q})) + \alpha_2 \cdot \sum_{t' \in H(\tilde{q})} Tr(t|t') \cdot \hat{p}(t'|H(\tilde{q})) + (1 - \alpha_1 - \alpha_2) \cdot \hat{p}(t|C) \quad (13)$$

$$p(t|T(\tilde{q})) = \beta_1 \cdot \hat{p}(t|T(\tilde{q})) + \beta_2 \cdot \sum_{t' \in T(\tilde{q})} Tr(t|t') \cdot \hat{p}(t'|T(\tilde{q})) + (1 - \beta_1 - \beta_2) \cdot \hat{p}(t|C) \quad (14)$$

where  $Tr(t|t')$  denotes the probability that topic term  $t$  is the translation of  $t'$ . In our experiments, to estimate the probability  $Tr(t|t')$ , we used the collections of question titles and question descriptions as the parallel corpus and the IBM model 1 (Brown et al., 1993) as the alignment model.

Usually, users reiterate or paraphrase their questions (already described in question titles) in question descriptions.

We utilized the new model elaborated by equation (13) and (14) for searching questions about ‘travel’ and ‘computers & internet’. The new model is denoted as ‘SMT-CUT’. Table 6 provides the evaluation results. The evaluation was conducted with exactly the same setting as in Section 3. From Table 6, we see that the performance of our approach can be further boosted by using translation probability.

Data	Methods	MAP	R-Precision	MRR
TRL-TST	LMIR-CUT	0.236	0.192	0.279
	SMT-CUT	<b>0.266</b>	<b>0.225</b>	<b>0.308</b>
CI-TST	LMIR-CUT	0.279	0.230	<b>0.341</b>
	SMT-CUT	<b>0.282</b>	<b>0.236</b>	0.337

Table 6. Using Translation Probability

## 5 Related Work

The major focus of previous research efforts on question search is to tackle the lexical chasm problem between questions.

The research of question search is first conducted using FAQ data. FAQ Finder (Burke et al., 1997) heuristically combines statistical similarities and semantic similarities between questions to rank FAQs. Conventional vector space models are used to calculate the statistical similarity and WordNet (Fellbaum, 1998) is used to estimate the semantic similarity. Sneyders (2002) proposed template based FAQ retrieval systems. Lai et al. (2002) proposed an approach to automatically mine FAQs from the Web. Jijkoun and Rijke (2005) used supervised learning methods to extend heuristic extraction of Q/A pairs from FAQ pages, and treated Q/A pair retrieval as a fielded search task.

Harabagiu et al. (2005) used a Question Answer Database (known as QUAB) to support interactive question answering. They compared seven different similarity metrics for selecting related questions from QUAB and found that the concept-based metric performed best.

Recently, the research of question search has been further extended to the community-based Q&A data. For example, Jeon et al. (Jeon et al., 2005a; Jeon et al., 2005b) compared four different retrieval methods, i.e. vector space model, Okapi, language model (LM), and translation-based model, for automatically fixing the lexical chasm between

questions of question search. They found that the translation-based model performed best.

However, all the existing methods treat questions just as plain texts (without considering question structure). In this paper, we proposed to conduct question search by identifying question topic and question focus. To the best of our knowledge, none of the existing studies addressed question search by modeling both question topic and question focus.

Question answering (e.g., Pasca and Harabagiu, 2001; Echihabi and Marcu, 2003; Voorhees, 2004; Metzler and Croft, 2005) relates to question search. Question answering automatically extracts short answers for a relatively limited class of question types from document collections. In contrast to that, question search retrieves answers for an unlimited range of questions by focusing on finding semantically similar questions in an archive.

## 6 Conclusions and Future Work

In this paper, we have proposed an approach to question search which models question topic and question focus in a language modeling framework.

The contribution of this paper can be summarized in 4-fold: (1) A data structure consisting of *question topic* and *question focus* was proposed for summarizing questions; (2) The MDL-based tree cut model was employed to identify question topic and question focus automatically; (3) A new form of language modeling using question topic and question focus was developed for question search; (4) Extensive experiments have been conducted to evaluate the proposed approach using a large collection of real questions obtained from Yahoo! Answers.

Though we only utilize data from community-based question answering service in our experiments, we could also use categorized questions from forum sites and FAQ sites. Thus, as future work, we will try to investigate the use of the proposed approach for other kinds of web services.

## Acknowledgement

We would like to thank Xinying Song, Shasha Li, and Shilin Ding for their efforts on developing the evaluation data. We would also like to thank Stephan H. Stiller for his proof-reading of the paper.

## References

- A. Echihabi and D. Marcu. 2003. A Noisy-Channel Approach to Question Answering. In *Proc. of ACL '03*.
- C. Fellbaum. 1998. WordNet: An electronic lexical database. *MIT Press*.
- D. Metzler and W. B. Croft. 2005. Analysis of statistical question classification for fact-based questions. *Information Retrieval*, 8(3), pages 481–504.
- E. Fredkin. 1960. Trie memory. *Communications of the ACM*, D. 3(9):490–499.
- E. M. Voorhees. 2004. Overview of the TREC 2004 question answering track. In *Proc. of TREC'04*.
- E. Sneiders. 2002. Automated question answering using question templates that cover the conceptual model of the database. In *Proc. of the 6th International Conference on Applications of Natural Language to Information Systems*, pages 235–239.
- G. Salton, A. Wong, and C. S. Yang 1975. A vector space model for automatic indexing. *Communications of the ACM*, vol. 18, nr. 11, pages 613–620.
- H. Li and N. Abe. 1998. Generalizing case frames using a thesaurus and the MDL principle. *Computational Linguistics*, 24(2), pages 217–244.
- J. Jeon and W.B. Croft. 2007. Learning translation-based language models using Q&A archives. Technical report, University of Massachusetts.
- J. Jeon, W. B. Croft, and J. Lee. 2005a. Finding semantically similar questions based on their answers. In *Proc. of SIGIR'05*.
- J. Jeon, W. B. Croft, and J. Lee. 2005b. Finding similar questions in large question and answer archives. In *Proc. of CIKM '05*, pages 84–90.
- J. Rissanen. 1978. Modeling by shortest data description. *Automatica*, vol. 14, pages. 465–471
- J.M. Ponte, W.B. Croft. 1998. A language modeling approach to information retrieval. In *Proc. of SIGIR'98*.
- M. A. Pasca and S. M. Harabagiu. 2001. High performance question/answering. In *Proc. of SIGIR'01*, pages 366–374.
- P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.
- R. D. Burke, K. J. Hammond, V. A. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg. 1997. Question answering from frequently asked question files: Experiences with the FAQ finder system. Technical report, University of Chicago.
- S. Harabagiu, A. Hickl, J. Lehmann and D. Moldovan. 2005. Experiments with Interactive Question-Answering. In *Proc. of ACL '05*.
- V. Jijkoun, M. D. Rijke. 2005. Retrieving Answers from Frequently Asked Questions Pages on the Web. In *Proc. of CIKM'05*.
- Y. Cao and H. Li. 2002. Base noun phrase translation using web data and the EM algorithm. In *Proc. of COLING '02*.
- Y.-S. Lai, K.-A. Fung, and C.-H. Wu. 2002. Faq mining via list detection. In *Proc. of the Workshop on Multilingual Summarization and Question Answering, 2002*.



# Trainable Generation of Big-Five Personality Styles through Data-driven Parameter Estimation

François Mairesse

Cambridge University Engineering Department  
Trumpington Street  
Cambridge, CB2 1PZ, United Kingdom  
farm2@eng.cam.ac.uk

Marilyn Walker

Department of Computer Science  
University of Sheffield  
Sheffield, S1 4DP, United Kingdom  
lynwalker@gmail.com

## Abstract

Previous work on statistical language generation has primarily focused on grammaticality and naturalness, scoring generation possibilities according to a language model or user feedback. More recent work has investigated data-driven techniques for controlling linguistic style without overgeneration, by reproducing variation dimensions extracted from corpora. Another line of work has produced handcrafted *rule-based* systems to control specific stylistic dimensions, such as politeness and personality. This paper describes a novel approach that automatically learns to produce recognisable variation along a meaningful stylistic dimension—personality—without the computational cost incurred by overgeneration techniques. We present the first evaluation of a data-driven generation method that projects multiple personality traits *simultaneously* and on a *continuous* scale. We compare our performance to a rule-based generator in the same domain.

## 1 Introduction

Over the last 20 years, statistical language models (SLMs) have been used successfully in many tasks in natural language processing, and the data available for modeling has steadily grown (Lapata and Keller, 2005). Langkilde and Knight (1998) first applied SLMs to statistical natural language generation (SNLG), showing that high quality paraphrases can be generated from an underspecified representation of meaning, by first applying a very underconstrained, rule-based *overgeneration* phase, whose outputs are then ranked by an SLM *scoring* phase. Since then, research in SNLG has explored a range of models for both dialogue and text generation.

One line of work has primarily focused on grammaticality and naturalness, scoring the overgener-

ation phase with a SLM, and evaluating against a gold-standard corpus, using string or tree-match metrics (Langkilde-Geary, 2002; Bangalore and Rambow, 2000; Chambers and Allen, 2004; Belz, 2005; Isard et al., 2006).

Another thread investigates SNLG scoring models trained using higher-level linguistic features to replicate human judgments of utterance quality (Rambow et al., 2001; Nakatsu and White, 2006; Stent and Guo, 2005). The error of these scoring models approaches the gold-standard human ranking with a relatively small training set.

A third SNLG approach eliminates the overgeneration phase (Paiva and Evans, 2005). It applies factor analysis to a corpus exhibiting stylistic variation, and then learns which generation parameters to manipulate to correlate with factor measurements. The generator was shown to reproduce intended factor levels across several factors, thus modelling the stylistic variation as measured in the original corpus.

Our goal is a generation technique that can target multiple stylistic effects *simultaneously* and over a *continuous* scale, controlling stylistic dimensions that are commonly understood and thus *meaningful* to users and application developers. Our intended applications are output utterances for intelligent training or intervention systems, video game characters, or virtual environment avatars. In previous work, we presented PERSONAGE, a psychologically-informed rule-based generator based on the Big Five personality model, and we showed that PERSONAGE can project extreme personality on the extraversion scale, i.e. both introverted and extraverted personality types (Mairesse and Walker, 2007). We used the Big Five model to develop PERSONAGE for several reasons. First, the Big Five has been shown in psychology to ex-

Trait	High	Low
<b>Extraversion</b>	warm, assertive, sociable, excitement seeking, active, spontaneous, optimistic, talkative	shy, quiet, reserved, passive, solitary, moody
<b>Emotional stability</b>	calm, even-tempered, reliable, peaceful, confident	neurotic, anxious, depressed, self-conscious
<b>Agreeableness</b>	trustworthy, considerate, friendly, generous, helpful	unfriendly, selfish, suspicious, uncooperative, malicious
<b>Conscientiousness</b>	competent, disciplined, dutiful, achievement striving	disorganised, impulsive, unreliable, forgetful
<b>Openness to experience</b>	creative, intellectual, curious, cultured, complex	narrow-minded, conservative, ignorant, simple

Table 1: Example adjectives associated with extreme values of the Big Five trait scales.

plain much of the variation in human perceptions of personality differences. Second, we believe that the adjectives used to develop the Big Five model provide an intuitive, *meaningful* definition of linguistic style. Table 1 shows some of the trait adjectives associated with the extremes of each Big Five trait. Third, there are many studies linking personality to linguistic variables (Pennebaker and King, 1999; Mehl et al., 2006, *inter alia*). See (Mairesse and Walker, 2007) for more detail.

In this paper, we further test the utility of basing stylistic variation on the Big Five personality model. The Big Five traits are represented by scalar values that range from 1 to 7, with values normally distributed among humans. While our previous work targeted extreme values of individual traits, here we show that we can target multiple personality traits *simultaneously* and over the *continuous* scales of the Big Five model. Section 2 describes a novel parameter-estimation method that automatically learns to produce recognisable variation for all Big Five traits, without overgeneration, implemented in a new SNLG called PERSONAGE-PE. We show that PERSONAGE-PE generates targets for multiple personality dimensions, using linear and non-linear parameter estimation models to predict generation parameters *directly* from the scalar targets. Section 3.2 shows that humans accurately perceive the intended variation, and Section 3.3 compares PERSONAGE-PE (trained) with PERSONAGE (rule-based; Mairesse and Walker, 2007). We delay a detailed discussion of related work to Section 4, where we summarize and discuss future work.

## 2 Parameter Estimation Models

The data-driven parameter estimation method consists of a development phase and a generation phase (Section 3). The development phase:

1. Uses a base generator to produce multiple utterances by randomly varying its parameters;
2. Collects human judgments rating the personality of each utterance;
3. Trains statistical models to predict the parameters from the personality judgments;

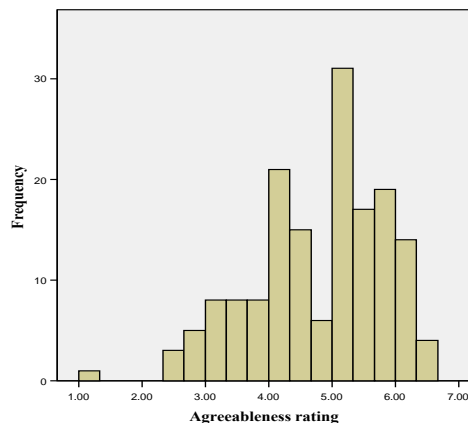


Figure 1: Distribution of average agreeableness ratings from the 2 expert judges for 160 random utterances.

4. Selects the best model for each parameter via cross-validation.

### 2.1 Base Generator

We make minimal assumptions about the input to the generator to favor domain independence. The input is a speech act, a potential content pool that can be used to achieve that speech act, and five scalar personality parameters (1..7), specifying values for the continuous scalar dimensions of each trait in the Big Five model. See Table 1. This requires a base generator that generates multiple outputs expressing the same input content by varying linguistic parameters related to the Big Five traits. We start with the PERSONAGE generator (Mairesse and Walker, 2007), which generates recommendations and comparisons of restaurants. We extend PERSONAGE with new parameters for a total of 67 parameters in PERSONAGE-PE. See Table 2. These parameters are derived from psychological studies identifying linguistic markers of the Big Five traits (Pennebaker and King, 1999; Mehl et al., 2006, *inter alia*). As PERSONAGE’s input parameters are domain-independent, most parameters range continuously between 0 and 1, while pragmatic marker insertion parameters are binary, except for the SUBJECT IMPLICITNESS, STUTTERING and PRONOMI-

Parameters	Description
<b>Content parameters:</b>	
VERBOSITY	Control the number of propositions in the utterance
RESTATEMENTS	Paraphrase an existing proposition, e.g. <i>'Chanpen Thai has great service, it has fantastic waiters'</i>
REPETITIONS	Repeat an existing proposition
CONTENT POLARITY	Control the polarity of the propositions expressed, i.e. referring to negative or positive attributes
REPETITIONS POLARITY	Control the polarity of the restated propositions
CONCESSIONS	Emphasise one attribute over another, e.g. <i>'even if Chanpen Thai has great food, it has bad service'</i>
CONCESSIONS POLARITY	Determine whether positive or negative attributes are emphasised
POLARISATION	Control whether the expressed polarity is neutral or extreme
POSITIVE CONTENT FIRST	Determine whether positive propositions—including the claim—are uttered first
<b>Syntactic template selection parameters:</b>	
SELF-REFERENCES	Control the number of first person pronouns
CLAIM COMPLEXITY	Control the syntactic complexity (syntactic embedding)
CLAIM POLARITY	Control the connotation of the claim, i.e. whether positive or negative affect is expressed
<b>Aggregation operations:</b>	
PERIOD	Leave two propositions in their own sentences, e.g. <i>'Chanpen Thai has great service. It has nice decor.'</i>
RELATIVE CLAUSE	Aggregate propositions with a relative clause, e.g. <i>'Chanpen Thai, which has great service, has nice decor'</i>
WITH CUE WORD	Aggregate propositions using <i>with</i> , e.g. <i>'Chanpen Thai has great service, with nice decor'</i>
CONJUNCTION	Join two propositions using a conjunction, or a comma if more than two propositions
MERGE	Merge the subject and verb of two propositions, e.g. <i>'Chanpen Thai has great service and nice decor'</i>
ALSO CUE WORD	Join two propositions using <i>also</i> , e.g. <i>'Chanpen Thai has great service, also it has nice decor'</i>
CONTRAST - CUE WORD	Contrast two propositions using <i>while, but, however, on the other hand</i> , e.g. <i>'While Chanpen Thai has great service, it has bad decor', 'Chanpen Thai has great service, but it has bad decor'</i>
JUSTIFY - CUE WORD	Justify a proposition using <i>because, since, so</i> , e.g. <i>'Chanpen Thai is the best, because it has great service'</i>
CONCEDE - CUE WORD	Concede a proposition using <i>although, even if, but/though</i> , e.g. <i>'Although Chanpen Thai has great service, it has bad decor', 'Chanpen Thai has great service, but it has bad decor though'</i>
MERGE WITH COMMA	Restate a proposition by repeating only the object, e.g. <i>'Chanpen Thai has great service, nice waiters'</i>
CONJ. WITH ELLIPSIS	Restate a proposition after replacing its object by an ellipsis, e.g. <i>'Chanpen Thai has . . . , it has great service'</i>
<b>Pragmatic markers:</b>	
SUBJECT IMPLICITNESS	Make the restaurant implicit by moving the attribute to the subject, e.g. <i>'the service is great'</i>
NEGATION	Negate a verb by replacing its modifier by its antonym, e.g. <i>'Chanpen Thai doesn't have bad service'</i>
SOFTENER HEDGES	Insert syntactic elements ( <i>sort of, kind of, somewhat, quite, around, rather, I think that, it seems to me that</i> ) to mitigate the strength of a proposition, e.g. <i>'Chanpen Thai has kind of great service'</i> or <i>'It seems to me that Chanpen Thai has rather great service'</i>
EMPHASIZER HEDGES	Insert syntactic elements ( <i>really, basically, actually, just</i> ) to strengthen a proposition, e.g. <i>'Chanpen Thai has really great service'</i> or <i>'Basically, Chanpen Thai just has great service'</i>
ACKNOWLEDGMENTS	Insert an initial back-channel ( <i>yeah, right, ok, I see, oh, well</i> ), e.g. <i>'Well, Chanpen Thai has great service'</i>
FILLED PAUSES	Insert syntactic elements expressing hesitancy ( <i>like, I mean, err, mmhm, you know</i> ), e.g. <i>'I mean, Chanpen Thai has great service, you know'</i> or <i>'Err... Chanpen Thai has, like, great service'</i>
EXCLAMATION	Insert an exclamation mark, e.g. <i>'Chanpen Thai has great service!'</i>
EXPLETIVES	Insert a swear word, e.g. <i>'the service is damn great'</i>
NEAR-EXPLETIVES	Insert a near-swear word, e.g. <i>'the service is darn great'</i>
COMPETENCE MITIGATION	Express the speaker's negative appraisal of the hearer's request, e.g. <i>'everybody knows that . . .'</i>
TAG QUESTION	Insert a tag question, e.g. <i>'the service is great, isn't it?'</i>
STUTTERING	Duplicate the first letters of a restaurant's name, e.g. <i>'Ch-ch-anpen Thai is the best'</i>
CONFIRMATION	Begin the utterance with a confirmation of the restaurant's name, e.g. <i>'did you say Chanpen Thai?'</i>
INITIAL REJECTION	Begin the utterance with a mild rejection, e.g. <i>'I'm not sure'</i>
IN-GROUP MARKER	Refer to the hearer as a member of the same social group, e.g. <i>pal, mate and buddy</i>
PRONOMINALIZATION	Replace occurrences of the restaurant's name by pronouns
<b>Lexical choice parameters:</b>	
LEXICAL FREQUENCY	Control the average frequency of use of each content word, according to BNC frequency counts
WORD LENGTH	Control the average number of letters of each content word
VERB STRENGTH	Control the strength of the selected verbs, e.g. <i>'I would suggest'</i> vs. <i>'I would recommend'</i>

Table 2: The 67 generation parameters whose target values are learned. Aggregation cue words, hedges, acknowledgments and filled pauses are learned individually (as separate parameters), e.g. *kind of* is modeled differently than *somewhat* in the SOFTENER HEDGES category. Parameters are detailed in previous work (Mairesse and Walker, 2007).

NALIZATION parameters.

## 2.2 Random Sample Generation and Expert Judgments

We generate a sample of 160 *random* utterances by varying the parameters in Table 2 with a uniform distribution. This sample is intended to provide enough training material for estimating all 67 parameters for each personality dimension. Following Mairesse

and Walker (2007), two expert judges (not the authors) familiar with the Big Five adjectives (Table 1) evaluate the personality of each utterance using the Ten-Item Personality Inventory (TIPI; Gosling et al., 2003), and also judge the utterance's naturalness. Thus 11 judgments were made for each utterance for a total of 1760 judgments. The TIPI outputs a rating on a scale from 1 (low) to 7 (high) for each Big Five trait. The expert judgments are approximately nor-

mally distributed; Figure 1 shows the distribution for agreeableness.

### 2.3 Statistical Model Training

Training data is created for each generation parameter—i.e. the output variable—to train statistical models predicting the optimal parameter value from the target personality scores. The models are thus based on the simplifying assumption that the generation parameters are independent. Any personality trait whose correlation with a generation decision is below 0.1 is removed from the training data. This has the effect of removing parameters that do not correlate strongly with any trait, which are set to a constant default value at generation time. Since the input parameter values may not be satisfiable depending on the input content, the actual generation decisions made for each utterance are recorded. For example, the CONCESSIONS decision value is the actual number of concessions produced in the utterance. To ensure that the models’ output can control the generator, the generation decision values are normalized to match the input range (0..1) of PERSONAGE-PE. Thus the dataset consists of 160 utterances and the corresponding generation decisions, each associated with 5 personality ratings averaged over both judges.

Parameter estimation models are trained to predict either continuous (e.g. VERBOSITY) or binary (e.g. EXCLAMATION) generation decisions. We compare various learning algorithms using the Weka toolkit (with default values unless specified; Witten and Frank, 2005). Continuous parameters are modeled with a linear regression model (LR), an M5’ model tree (M5), and a model based on support vector machines with a linear kernel (SVM). As regression models can extrapolate beyond the [0, 1] interval, the output parameter values are truncated if needed—at generation time—before being sent to the base generator. Binary parameters are modeled using classifiers that predict whether the parameter is *enabled* or *disabled*. We test a Naive Bayes classifier (NB), a j48 decision tree (J48), a nearest-neighbor classifier using one neighbor (NN), a Java implementation of the RIPPER rule-based learner (JRIP), the AdaBoost boosting algorithm (ADA), and a support vector machines classifier with a linear kernel (SVM).

Figures 2, 3 and 4 show the models learned for the EXCLAMATION (binary), STUTTERING (continuous), and CONTENT POLARITY (continuous) parameters in Table 2. The models predict generation parameters from input personality scores; note that

Condition	Class	Weight
if extraversion > 6.42 then 1 else 0	1	1.81
if extraversion > 4.42 then 1 else 0	1	0.38
if extraversion <= 6.58 then 1 else 0	1	0.22
if extraversion > 4.71 then 1 else 0	1	0.28
if agreeableness > 5.13 then 1 else 0	1	0.42
if extraversion <= 6.58 then 1 else 0	1	0.14
if extraversion > 4.79 then 1 else 0	1	0.19
if extraversion <= 6.58 then 1 else 0	1	0.17

Figure 2: AdaBoost model predicting the EXCLAMATION parameter. Given input trait values, the model outputs the class yielding the largest sum of weights for the rules returning that class. Class 0 = *disabled*, class 1 = *enabled*.

(normalized) Content polarity =
0.054
- 0.102 * (normalized) emotional stability
+ 0.970 * (normalized) agreeableness
- 0.110 * (normalized) conscientiousness
+ 0.013 * (normalized) openness to experience

Figure 3: SVM model with a linear kernel predicting the CONTENT POLARITY parameter.

sometimes the best performing model is non-linear. Given input trait values, the AdaBoost model in Figure 2 outputs the class yielding the largest sum of weights for the rules returning that class. For example, the first rule of the EXCLAMATION model shows that an extraversion score above 6.42 out of 7 would increase the weight of the *enabled* class by 1.81. The fifth rule indicates that a target agreeableness above 5.13 would further increase the weight by .42. The STUTTERING model tree in Figure 4 lets us calculate that a low emotional stability (1.0) together with a neutral conscientiousness and openness to experience (4.0) yield a parameter value of .62 (see LM2), whereas a neutral emotional stability decreases the value down to .17. Figure 4 also shows how personality traits that do not affect the parameter are removed, i.e. emotional stability, conscientiousness and openness to experience are the traits that affect stuttering. The linear model in Figure 3 shows that agreeableness has a strong effect on the CONTENT POLARITY parameter (.97 weight), but emotional stability, conscientiousness and openness to experience also have an effect.

### 2.4 Model Selection

The final step of the development phase identifies the best performing model(s) for each generation parameter via cross-validation. For continuous pa-

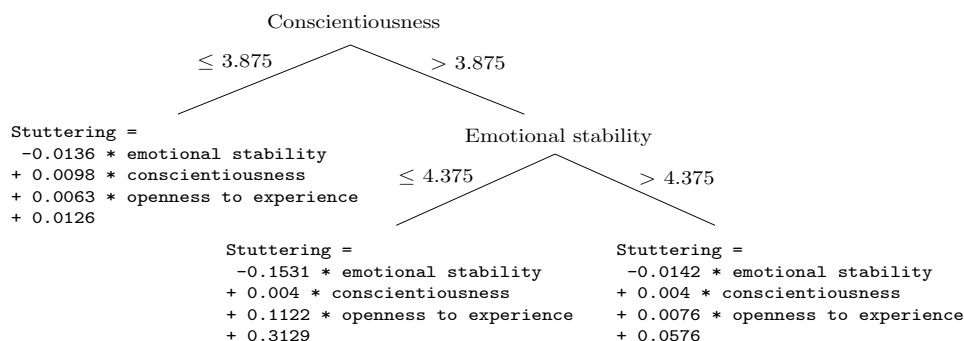


Figure 4: M5' model tree predicting the STUTTERING parameter.

Continuous parameters	LR	M5	SVM
<b>Content parameters:</b>			
VERBOSITY	0.24	<b>0.26</b>	0.21
RESTATEMENTS	0.14	<b>0.14</b>	0.04
REPETITIONS	<b>0.13</b>	0.13	0.08
CONTENT POLARITY	0.46	0.46	<b>0.47</b>
REPETITIONS POLARITY	0.02	<b>0.15</b>	0.06
CONCESSIONS	<b>0.23</b>	0.23	0.12
CONCESSIONS POLARITY	-0.01	<b>0.16</b>	0.07
POLARISATION	0.20	<b>0.21</b>	0.20
<b>Syntactic template selection:</b>			
CLAIM COMPLEXITY	0.10	<b>0.33</b>	0.26
CLAIM POLARITY	0.04	0.04	<b>0.05</b>
<b>Aggregation operations:</b>			
INFER - WITH CUE WORD	0.03	<b>0.03</b>	0.01
INFER - ALSO CUE WORD	<b>0.10</b>	0.10	0.06
JUSTIFY - SINCE CUE WORD	0.03	<b>0.07</b>	0.05
JUSTIFY - SO CUE WORD	0.07	<b>0.07</b>	0.04
JUSTIFY - PERIOD	<b>0.36</b>	0.35	0.21
CONTRAST - PERIOD	<b>0.27</b>	0.26	0.26
RESTATE - MERGE WITH COMMA	0.18	<b>0.18</b>	0.09
CONCEDE - ALTHOUGH CUE WORD	<b>0.08</b>	0.08	0.05
CONCEDE - EVEN IF CUE WORD	0.05	<b>0.05</b>	0.03
<b>Pragmatic markers:</b>			
SUBJECT IMPLICITNESS	<b>0.13</b>	0.13	0.04
STUTTERING INSERTION	0.16	<b>0.23</b>	0.17
PRONOMINALIZATION	<b>0.22</b>	0.20	0.17
<b>Lexical choice parameters:</b>			
LEXICAL FREQUENCY	0.21	<b>0.21</b>	0.19
WORD LENGTH	<b>0.18</b>	0.18	0.15

Table 3: Pearson’s correlation between parameter model predictions and continuous parameter values, for different regression models. Parameters that do not correlate with any trait are omitted. Aggregation operations are associated with a rhetorical relation (e.g. INFER). Results are averaged over a 10-fold cross-validation.

rameters, Table 3 evaluates modeling accuracy by comparing the correlations between the model’s predictions and the actual parameter values in the test folds. Table 4 reports results for binary parameter classifiers, by comparing the F-measures of the *enabled* class. Best performing models are identified in bold; parameters that do not correlate with any trait or that produce a poor modeling accuracy are omitted.

The CONTENT POLARITY parameter is modeled

Binary parameters	NB	J48	NN	ADA	SVM
<b>Pragmatic markers:</b>					
<b>SOFTENER HEDGES</b>					
<i>kind of</i>	0.00	0.00	<b>0.16</b>	0.11	0.10
<i>rather</i>	0.00	0.00	<b>0.02</b>	0.01	0.01
<i>quite</i>	<b>0.14</b>	0.08	0.09	0.07	0.06
<b>EMPHASIZER HEDGES</b>					
<i>basically</i>	0.00	0.00	<b>0.02</b>	0.01	0.01
<b>ACKNOWLEDGMENTS</b>					
<i>yeah</i>	0.00	0.00	<b>0.04</b>	0.03	0.03
<i>ok</i>	<b>0.13</b>	0.07	0.06	0.05	0.05
<b>FILLED PAUSES</b>					
<i>err</i>	<b>0.32</b>	0.20	0.24	0.22	0.19
EXCLAMATION	0.23	0.34	0.36	<b>0.38</b>	0.34
EXPLETIVES	<b>0.27</b>	0.18	0.24	0.17	0.15
IN-GROUP MARKER	<b>0.40</b>	0.31	0.31	0.24	0.21
TAG QUESTION	<b>0.32</b>	0.21	0.21	0.15	0.13
CONFIRMATION	0.00	0.00	<b>0.07</b>	0.04	0.04

Table 4: F-measure of the *enabled* class for classification models of binary parameters. Parameters that do not correlate with any trait are omitted. Results are averaged over a 10-fold cross-validation. JRIP models are not shown as they never perform best.

the most accurately, with the SVM model in Figure 3 producing a correlation of .47 with the true parameter values. Models of the PERIOD aggregation operation also perform well, with a linear regression model yielding a correlation of .36 when realizing a justification, and .27 when contrasting two propositions. CLAIM COMPLEXITY and VERBOSITY are also modeled successfully, with correlations of .33 and .26 using a model tree. The model tree controlling the STUTTERING parameter illustrated in Figure 4 produces a correlation of .23. For binary parameters, Table 4 shows that the Naive Bayes classifier is generally the most accurate, with F-measures of .40 for the IN-GROUP MARKER parameter, and .32 for both the insertion of filled pauses (*err*) and tag questions. The AdaBoost algorithm best predicts the EXCLAMATION parameter, with an F-measure of .38 for the model in Figure 2.

#	Traits	End	Rating	Nat	Output utterance
1.a	Extraversion Agreeableness	high high	4.42 4.94	4.79	Radio Perfecto’s price is 25 dollars but Les Routiers provides adequate food. I imagine they’re alright!
1.b	Emotional stability Conscientiousness	high high	5.35 5.21	5.04	Let’s see, Les Routiers and Radio Perfecto... You would probably appreciate them. Radio Perfecto is in the East Village with kind of acceptable food. Les Routiers is located in Manhattan. Its price is 41 dollars.
2.a	Extraversion Agreeableness	low low	3.65 4.02	3.21	Err... you would probably appreciate Trattoria Rustica, wouldn’t you? It’s in Manhattan, also it’s an italian restaurant. It offers poor ambience, also it’s quite costly.
2.b	Emotional stability Openness to experience	low low	4.13 3.85	4.50	Trattoria Rustica isn’t as bad as the others. Err... even if it’s costly, it offers kind of adequate food, alright? It’s an italian place.

Table 5: Example outputs controlled by the parameter estimation models for a comparison (#1) and a recommendation (#2), with the average judges’ ratings (*Rating*) and naturalness (*Nat*). Ratings are on a scale from 1 to 7, with 1 = very low (e.g. neurotic or introvert) and 7 = very high on the dimension (e.g. emotionally stable or extraverted).

### 3 Evaluation Experiment

The generation phase of our parameter estimation SNLG method consists of the following steps:

1. Use the best performing models to predict parameter values from the desired personality scores;
2. Generate the output utterance using the predicted parameter values.

We then evaluate the output utterances using naive human judges to rate their perceived personality and naturalness.

#### 3.1 Evaluation Method

Given the best performing model for each generation parameter, we generate 5 utterances for each of 5 recommendation and 5 comparison speech acts. Each utterance targets an extreme value for two traits (either 1 or 7 out of 7) and neutral values for the remaining three traits (4 out of 7). The goal is for each utterance to project *multiple* traits on a *continuous* scale. To generate a range of alternatives, a Gaussian noise with a standard deviation of 10% of the full scale is added to each target value.

Subjects were 24 native English speakers (12 male and 12 female graduate students from a range of disciplines from both the U.K. and the U.S.). Subjects evaluate the naturalness and personality of each utterance using the TIPI (Gosling et al., 2003). To limit the experiment’s duration, only the two traits with extreme target values are evaluated for each utterance. Subjects thus answered 5 questions for 50 utterances, two from the TIPI for each extreme trait and one about naturalness (250 judgments in total per subject). Subjects were not told that the utterances were intended to manifest extreme trait values. Table 5 shows several sample outputs and the mean personality ratings from the human judges. For example, utterance 1.a projects a high extraversion through the insertion of an exclamation mark

based on the model in Figure 2, whereas utterance 2.a conveys introversion by beginning with the filled pause *err*. The same utterance also projects a low agreeableness by focusing on negative propositions, through a low CONTENT POLARITY parameter value as per the model in Figure 3. This evaluation addresses a number of open questions discussed below.

- Q1: Is the personality projected by models trained on ratings from a few expert judges recognised by a larger sample of naive judges? (Section 3.2)
- Q2: Can a *combination* of multiple traits within a single utterance be detected by naive judges? (Section 3.2)
- Q3: How does PERSONAGE-PE compare to PERSONAGE, a psychologically-informed rule-based generator for projecting extreme personality? (Section 3.3)
- Q4: Does the parameter estimation SNLG method produce natural utterances? (Section 3.4)

#### 3.2 Parameter Estimation Evaluation

Table 6 shows that extraversion is the dimension modeled most accurately by the parameter estimation models, producing a .45 correlation with the subjects’ ratings ( $p < .01$ ). Emotional stability, agreeableness, and openness to experience ratings also correlate strongly with the target scores, with correlations of .39, .36 and .17 respectively ( $p < .01$ ). Additionally, Table 6 shows that the magnitude of the correlation increases when considering the perception of a hypothetical average subject, i.e. smoothing individual variation by averaging the ratings over all 24 judges, producing a correlation  $r_{avg}$  up to .80 for extraversion. These correlations are unexpectedly high; in corpus analyses, significant correlations as low as .05 to .10 are typically observed between personality and linguistic markers (Pennebaker and King, 1999; Mehl et al., 2006).

Conscientiousness is the only dimension whose ratings do not correlate with the target scores. The

comparison with rule-based results in Section 3.3 suggests that this is not because conscientiousness cannot be exhibited in our domain or manifested in a single utterance, so perhaps this arises from differing perceptions of conscientiousness between the expert and naive judges.

Trait	$r$	$r_{avg}$	$e$
Extraversion	.45 •	.80 •	1.89
Emotional stability	.39 •	.64 •	2.14
Agreeableness	.36 •	.68 •	2.38
Conscientiousness	-.01	-.02	2.79
Openness to experience	.17 •	.41 •	2.51

• statistically significant correlation  
 $p < .05$ , •  $p = .07$  (two-tailed)

Table 6: Pearson’s correlation coefficient  $r$  and mean absolute error  $e$  between the target personality scores and the 480 judges’ ratings (20 ratings per trait for 24 judges);  $r_{avg}$  is the correlation between the personality scores and the average judges’ ratings.

Table 6 shows that the mean absolute error varies between 1.89 and 2.79 on a scale from 1 to 7. Such large errors result from the decision to ask judges to answer just the TIPI questions for the two traits that were the extreme targets (See Section 3.1), because the judges tend to use the whole scale, with approximately normally distributed ratings. This means that although the judges make distinctions leading to high correlations, they do so on a compressed scale. This explains the large correlations despite the magnitude of the absolute error.

Table 7 shows results evaluating whether utterances targeting the extremes of a trait are perceived differently. The ratings differ significantly for all traits but conscientiousness ( $p \leq .001$ ). Thus parameter estimation models can be used in applications that only require discrete binary variation.

Trait	Low	High
Extraversion	3.69	5.06 •
Emotional stability	3.75	4.75 •
Agreeableness	3.42	4.33 •
Conscientiousness	4.16	4.15
Openness to experience	3.71	4.06 •

• statistically significant difference  
 $p \leq .001$  (two-tailed)

Table 7: Average personality ratings for the utterances generated with the low and high target values for each trait on a scale from 1 to 7.

It is important to emphasize that generation parameters were predicted based on 5 target personality values. Thus, the results show that *individual* traits are perceived even when utterances project

other traits as well, confirming that the Big Five theory models independent dimensions and thus provides a useful and meaningful framework for modeling variation in language. Additionally, although we do not directly evaluate the perception of mid-range values of personality target scores, the results suggest that mid-range personality is modeled correctly because the neutral target scores do not affect the perception of extreme traits.

### 3.3 Comparison with Rule-Based Generation

PERSONAGE is a rule-based personality generator based on handcrafted parameter settings derived from psychological studies. Mairesse and Walker (2007) show that this approach generates utterances that are perceptibly different along the extraversion dimension. Table 8 compares the mean ratings of the utterances generated by PERSONAGE-PE with ratings of 20 utterances generated by PERSONAGE for each extreme of each Big Five scale (40 for extraversion, resulting in 240 handcrafted utterances in total). Table 8 shows that the handcrafted parameter settings project a significantly more extreme personality for 6 traits out of 10. However, the learned parameter models for neuroticism, disagreeableness, unconscientiousness and openness to experience do not perform significantly worse than the handcrafted generator. These findings are promising as we discuss further in Section 4.

Method	Rule-based		Learned parameters	
	Low	High	Low	High
Extraversion	2.96	5.98	3.69 ◦	5.05 ◦
Emotional stability	3.29	5.96	3.75	4.75 ◦
Agreeableness	3.41	5.66	3.42	4.33 ◦
Conscientiousness	3.71	5.53	4.16	4.15 ◦
Openness to experience	2.89	4.21	3.71 ◦	4.06

◦, ◦ significant increase or decrease of the variation range over the average rule-based ratings ( $p < .05$ , two-tailed)

Table 8: Pair-wise comparison between the ratings of the utterances generated using PERSONAGE-PE with extreme target values (*Learned Parameters*), and the ratings for utterances generated with Mairesse and Walker’s rule-based PERSONAGE generator, (*Rule-based*). Ratings are averaged over all judges.

### 3.4 Naturalness Evaluation

The naive judges also evaluated the naturalness of the outputs of our trained models. Table 9 shows that the average naturalness is 3.98 out of 7, which is significantly lower ( $p < .05$ ) than the naturalness of handcrafted and randomly generated utterances reported by Mairesse and Walker (2007). It is possible that the differences arise from judgments of utterances targeting multiple traits, or that the naive

judges are more critical.

Trait	Rule-based	Random	Learned
All	4.59	4.38	3.98

Table 9: Average naturalness ratings for utterances generated using (1) PERSONAGE, the rule-based generator, (2) the random utterances (expert judges) and (3) the outputs of PERSONAGE-PE using the parameter estimation models (*Learned*, naive judges). The means differ significantly at the  $p < .05$  level (two-tailed independent sample t-test).

## 4 Conclusion

We present a new method for generating linguistic variation projecting multiple personality traits continuously, by combining and extending previous research in statistical natural language generation (Paiva and Evans, 2005; Rambow et al., 2001; Isard et al., 2006; Mairesse and Walker, 2007). While handcrafted rule-based approaches are limited to variation along a small number of discrete points (Hovy, 1988; Walker et al., 1997; Lester et al., 1997; Power et al., 2003; Cassell and Bickmore, 2003; Piwek, 2003; Mairesse and Walker, 2007; Rehm and André, in press), we learn models that predict parameter values for any arbitrary value on the variation dimension scales. Additionally, our data-driven approach can be applied to any dimension that is meaningful to human judges, and it provides an elegant way to project multiple dimensions simultaneously, by including the relevant dimensions as features of the parameter models’ training data.

Isard et al. (2006) and Mairesse and Walker (2007) also propose a personality generation method, in which a data-driven personality model selects the best utterance from a large candidate set. Isard et al.’s technique has not been evaluated, while Mairesse and Walker’s overgenerate and score approach is inefficient. Paiva and Evans’ technique does not overgenerate (2005), but it requires a search for the optimal generation decisions according to the learned models. Our approach does not require any search or overgeneration, as parameter estimation models predict the generation decisions directly from the target variation dimensions. This technique is therefore beneficial for real-time generation. Moreover the variation dimensions of Paiva and Evans’ data-driven technique are extracted from a corpus: there is thus no guarantee that they can be easily interpreted by humans, and that they generalise to other corpora. Previous work has shown that modeling the relation between personality and

language is far from trivial (Pennebaker and King, 1999; Argamon et al., 2005; Oberlander and Nowson, 2006; Mairesse et al., 2007), suggesting that the control of personality is a harder problem than the control of data-driven variation dimensions.

We present the first human perceptual evaluation of a data-driven stylistic variation method. In terms of our research questions in Section 3.1, we show that models trained on expert judges to project multiple traits in a single utterance generate utterances whose personality is recognized by naive judges. There is only one other similar evaluation of an SNLG (Rambow et al., 2001). Our models perform only slightly worse than a handcrafted rule-based generator in the same domain. These findings are promising as (1) parameter estimation models are able to target any combination of traits over the full range of the Big Five scales; (2) they do not benefit from psychological knowledge, i.e. they are trained on randomly generated utterances.

This work also has several limitations that should be addressed in future work. Even though the parameters of PERSONAGE-PE were suggested by psychological studies (Mairesse and Walker, 2007), some of them are not modeled successfully by our approach, and thus omitted from Tables 3 and 4. This could be due to the relatively small development dataset size (160 utterances to optimize 67 parameters), or to the implementation of some parameters. The strong parameter-independence assumption could also be responsible, but we are not aware of any state of the art implementation for learning multiple dependent variables, and this approach could further aggravate data sparsity issues.

In addition, it is unclear why PERSONAGE performs better for projecting extreme personality and produces more natural utterances, and why PERSONAGE-PE fails to project conscientiousness correctly. It might be possible to improve the parameter estimation models with a larger sample of random utterances at development time, or with additional extreme data generated using the rule-based approach. Such hybrid models are likely to perform better for extreme target scores, as they are trained on more uniformly distributed ratings (e.g. compared to the normal distribution in Figure 1). In addition, we have only shown that personality can be expressed by information presentation speech-acts in the restaurant domain; future work should assess the extent to which the parameters derived from psychological findings are culture, domain, and speech act dependent.



## References

- S. Argamon, S. Dhawle, M. Koppel, and J. Pennebaker. Lexical predictors of personality type. In *Proceedings of the Joint Annual Meeting of the Interface and the Classification Society of North America*, 2005.
- S. Bangalore and O. Rambow. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 42–48, 2000.
- A. Belz. Corpus-driven generation of weather forecasts. In *Proceedings of the 3rd Corpus Linguistics Conference*, 2005.
- J. Cassell and T. Bickmore. Negotiated collusion: Modeling social language and its relationship effects in intelligent agents. *User Modeling and User-Adapted Interaction*, 13:89–132, 2003.
- N. Chambers and J. Allen. Stochastic language generation in a dialogue system: Toward a domain independent generator. In *Proceedings 5th SIGdial Workshop on Discourse and Dialogue*, 2004.
- S. D. Gosling, P. J. Rentfrow, and W. B. Swann. A very brief measure of the big five personality domains. *Journal of Research in Personality*, 37:504–528, 2003.
- E. Hovy. *Generating Natural Language under Pragmatic Constraints*. Lawrence Erlbaum Associates, 1988.
- A. Isard, C. Brockmann, and J. Oberlander. Individuality and alignment in generated dialogues. In *Proceedings of the 4th International Natural Language Generation Conference (INLG)*, pages 22–29, 2006.
- I. Langkilde and K. Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 704–710, 1998.
- I. Langkilde-Geary. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 1st International Conference on Natural Language Generation*, 2002.
- M. Lapata and F. Keller. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2:1–31, 2005.
- J. Lester, S. Converse, S. Kahler, S. Barlow, B. Stone, and R. Bhogal. The persona effect: affective impact of animated pedagogical agents. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 359–366, 1997.
- F. Mairesse and M. A. Walker. PERSONAGE: Personality generation for dialogue. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 496–503, 2007.
- F. Mairesse, M. A. Walker, M. R. Mehl, and R. K. Moore. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of Artificial Intelligence Research (JAIR)*, 30:457–500, 2007.
- M. R. Mehl, S. D. Gosling, and J. W. Pennebaker. Personality in its natural habitat: Manifestations and implicit folk theories of personality in daily life. *Journal of Personality and Social Psychology*, 90:862–877, 2006.
- C. Nakatsu and M. White. Learning to say it well: Reranking realizations by predicted synthesis quality. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1113–1120, 2006.
- J. Oberlander and S. Nowson. Whose thumb is it anyway? classifying author personality from weblog text. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.
- D. S. Paiva and R. Evans. Empirically-based control of natural language generation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 58–65, 2005.
- J. W. Pennebaker and L. A. King. Linguistic styles: Language use as an individual difference. *Journal of Personality and Social Psychology*, 77:1296–1312, 1999.
- P. Piwek. A flexible pragmatics-driven language generator for animated agents. In *Proceedings of Annual Meeting of the European Chapter of the Association for Computational Linguistics (EACL)*, 2003.
- R. Power, D. Scott, and N. Bouayad-Agha. Generating texts with style. In *Proceedings of the 4th International Conference on Intelligent Text Processing and Computational Linguistics*, 2003.
- O. Rambow, M. Rogati, and M. A. Walker. Evaluating a trainable sentence planner for a spoken dialogue travel system. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2001.
- M. Rehm and E. André. From annotated multimodal corpora to simulated human-like behaviors. In I. Wachsmuth and G. Knoblich, editors, *Modeling Communication with Robots and Virtual Humans*. Springer, Berlin, Heidelberg, in press.
- A. Stent and H. Guo. A new data-driven approach for multimedia presentation generation. In *Proc. EuroIMSA*, 2005.
- M. A. Walker, J. E. Cahn, and S. J. Whittaker. Improvising linguistic style: Social and affective bases for agent personality. In *Proceedings of the 1st Conference on Autonomous Agents*, pages 96–105, 1997.
- I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, CA, 2005.

# Correcting Misuse of Verb Forms

John Lee and Stephanie Seneff

Spoken Language Systems

MIT Computer Science and Artificial Intelligence Laboratory

Cambridge, MA 02139, USA

{jsylee,seneff}@csail.mit.edu

## Abstract

This paper proposes a method to correct English verb form errors made by non-native speakers. A basic approach is template matching on parse trees. The proposed method improves on this approach in two ways. To improve recall, irregularities in parse trees caused by verb form errors are taken into account; to improve precision,  $n$ -gram counts are utilized to filter proposed corrections. Evaluation on non-native corpora, representing two genres and mother tongues, shows promising results.

## 1 Introduction

In order to describe the nuances of an action, a verb may be associated with various concepts such as tense, aspect, voice, mood, person and number. In some languages, such as Chinese, the verb itself is not inflected, and these concepts are expressed via other words in the sentence. In highly inflected languages, such as Turkish, many of these concepts are encoded in the inflection of the verb. In between these extremes, English uses a combination of inflections (see Table 1) and “helping words”, or auxiliaries, to form complex verb phrases.

It should come as no surprise, then, that the misuse of verb forms is a common error category for some non-native speakers of English. For example, in the Japanese Learners of English corpus (Izumi et al., 2003), errors related to verbs are among the most frequent categories. Table 2 shows some sentences with these errors.

Form	Example
base (bare)	<i>speak</i>
base (infinitive)	<i>to speak</i>
third person singular	<i>speaks</i>
past	<i>spoke</i>
-ing participle	<i>speaking</i>
-ed participle	<i>spoken</i>

Table 1: Five forms of inflections of English verbs (Quirk et al., 1985), illustrated with the verb “*speak*”. The base form is also used to construct the infinitive with “*to*”. An exception is the verb “*to be*”, which has more forms.

A system that automatically detects and corrects misused verb forms would be both an educational and practical tool for students of English. It may also potentially improve the performance of machine translation and natural language generation systems, especially when the source and target languages employ very different verb systems.

Research on automatic grammar correction has been conducted on a number of different parts-of-speech, such as articles (Knight and Chander, 1994) and prepositions (Chodorow et al., 2007). Errors in verb forms have been covered as part of larger systems such as (Heidorn, 2000), but we believe that their specific research challenges warrant more detailed examination.

We build on the basic approach of template-matching on parse trees in two ways. To improve recall, irregularities in parse trees caused by verb form errors are considered; to improve precision,  $n$ -gram counts are utilized to filter proposed corrections.

We start with a discussion on the scope of our

task in the next section. We then analyze the specific research issues in §3 and survey previous work in §4. A description of our data follows. Finally, we present experimental results and conclude.

## 2 Background

An English verb can be inflected in five forms (see Table 1). Our goal is to correct confusions among these five forms, as well as the infinitive. These confusions can be viewed as symptoms of one of two main underlying categories of errors; roughly speaking, one category is semantic in nature, and the other, syntactic.

### 2.1 Semantic Errors

The first type of error is concerned with inappropriate choices of tense, aspect, voice, or mood. These may be considered errors in semantics. In the sentence below, the verb “live” is expressed in the simple present tense, rather than the perfect progressive:

*He \*lives there since June.* (1)

Either “has been living” or “had been living” may be the valid correction, depending on the context. If there is no temporal expression, correction of tense and aspect would be even more challenging.

Similarly, correcting voice and mood often requires real-world knowledge. Suppose one wants to say “I am prepared for the exam”, but writes “I am preparing for the exam”. Semantic analysis of the context would be required to correct this kind of error, which will not be tackled in this paper<sup>1</sup>.

<sup>1</sup>If the input is “I am \*prepare for the exam”, however, we will attempt to choose between the two possibilities.

Example	Usage
<i>I take a bath and *reading books.</i>	FINITE
<i>I can't *skiing well, but ...</i>	BASE <sub>md</sub>
<i>Why did this *happened?</i>	BASE <sub>do</sub>
<i>But I haven't *decide where to go.</i>	ED <sub>perf</sub>
<i>I don't want *have a baby.</i>	INF <sub>verb</sub>
<i>I have to save my money for *ski.</i>	ING <sub>prep</sub>
<i>My son was very *satisfy with ...</i>	ED <sub>pass</sub>
<i>I am always *talk to my father.</i>	ING <sub>prog</sub>

Table 2: Sentences with verb form errors. The intended usages, shown on the right column, are defined in Table 3.

### 2.2 Syntactic Errors

The second type of error is the misuse of verb forms. Even if the intended tense, aspect, voice and mood are correct, the verb phrase may still be constructed erroneously. This type of error may be further subdivided as follows:

**Subject-Verb Agreement** The verb is not correctly inflected in number and person with respect to the subject. A common error is the confusion between the base form and the third person singular form, e.g.,

*He \*have been living there since June.* (2)

**Auxiliary Agreement** In addition to the modal auxiliaries, other auxiliaries must be used when specifying the perfective or progressive aspect, or the passive voice. Their use results in a complex verb phrase, i.e., one that consists of two or more verb constituents. Mistakes arise when the main verb does not “agree” with the auxiliary. In the sentence below, the present perfect progressive tense (“has been living”) is intended, but the main verb “live” is mistakenly left in the base form:

*He has been \*live there since June.* (3)

In general, the auxiliaries can serve as a hint to the intended verb form, even as the auxiliaries “has been” in the above case suggest that the progressive aspect was intended.

**Complementation** A nonfinite clause can serve as complementation to a verb or to a preposition. In the former case, the verb form in the clause is typically an infinitive or an -ing participle; in the latter, it is usually an -ing participle. Here is an example of a wrong choice of verb form in complementation to a verb:

*He wants \*live there.* (4)

In this sentence, “live”, in its base form, should be modified to its infinitive form as a complementation to the verb “wants”.

This paper focuses on correcting the above three error types: subject-verb agreement, auxiliary agreement, and complementation. Table 3 gives a complete list of verb form usages which will be covered.

Form	Usage	Description	Example
Base Form as Bare Infinitive	BASE <sub>md</sub> BASE <sub>do</sub>	After modals “Do”-support/-periphrasis; emphatic positive	He <b>may</b> call. <b>May</b> he call? He <b>did</b> not call. <b>Did</b> he call? I <b>did</b> call.
Base or 3rd person	FINITE	Simple present or past tense	He calls.
Base Form as to-Infinitive	INF <sub>verb</sub>	Verb complementation	He <b>wants</b> her to call.
-ing participle	ING <sub>prog</sub> ING <sub>verb</sub> ING <sub>prep</sub>	Progressive aspect Verb complementation Prepositional complementation	He <b>was</b> calling. <b>Was</b> he calling? He <b>hated</b> calling. The device is designed <b>for</b> calling
-ed participle	ED <sub>perf</sub> ED <sub>pass</sub>	Perfect aspect Passive voice	He <b>has</b> called. <b>Has</b> he called? He <b>was</b> called. <b>Was</b> he called?

Table 3: Usage of various verb forms. In the examples, the *italized* verbs are the “targets” for correction. In complementations, the main verbs or prepositions are **bolded**; in all other cases, the auxiliaries are **bolded**.

### 3 Research Issues

One strategy for correcting verb form errors is to identify the intended syntactic relationships between the verb in question and its neighbors. For subject-verb agreement, the subject of the verb is obviously crucial (e.g., “*he*” in (2)); the auxiliary is relevant for resolving auxiliary agreement (e.g., “*has been*” in (3)); determining the verb that receives the complementation is necessary for detecting any complementation errors (e.g., “*wants*” in (4)). Once these items are identified, most verb form errors may be corrected in a rather straightforward manner.

The success of this strategy, then, hinges on accurate identification of these items, for example, from parse trees. Ambiguities will need to be resolved, leading to two research issues (§3.2 and §3.3).

#### 3.1 Ambiguities

The three so-called *primary verbs*, “*have*”, “*do*” and “*be*”, can serve as either main or auxiliary verbs. The verb “*be*” can be utilized as a main verb, but also as an auxiliary in the progressive aspect (ING<sub>prog</sub> in Table 3) or the passive voice (ED<sub>pass</sub>). The three examples below illustrate these possibilities:

*This is work not play.* (main verb)  
*My father is working in the lab.* (ING<sub>prog</sub>)  
*A solution is worked out.* (ED<sub>pass</sub>)

ambiguity among these roles is usually straightforward because of the different verb forms (e.g., “*working*” vs. “*worked*”). If the verb forms are incorrect, disambiguation is made more difficult:

*This is work not play.*  
*My father is \*work in the lab.*  
*A solution is \*work out.*

Similar ambiguities are introduced by the other primary verbs<sup>2</sup>. The verb “*have*” can function as an auxiliary in the perfect aspect (ED<sub>perf</sub>) as well as a main verb. The versatile “*do*” can serve as “do”-support or add emphasis (BASE<sub>do</sub>), or simply act as a main verb.

#### 3.2 Automatic Parsing

The ambiguities discussed above may be expected to cause degradation in automatic parsing performance. In other words, sentences containing verb form errors are more likely to yield an “incorrect” parse tree, sometimes with significant differences. For example, the sentence “*My father is \*work in the laboratory*” is parsed (Collins, 1997) as:

(S (NP My father)  
(VP is (NP work))  
(PP in the laboratory))

These different roles clearly affect the forms required for the verbs (if any) that follow. Dis-

<sup>2</sup>The abbreviations ‘s (*is* or *has*) and ‘d (*would* or *had*) compound the ambiguities.

The progressive form “*working*” is substituted with its bare form, which happens to be also a noun. The parser, not unreasonably, identifies “*work*” as a noun. Correcting the *verb* form error in this sentence, then, necessitates considering the *noun* that is apparently a copular complementation.

Anecdotal observations like this suggest that one cannot use parser output naively<sup>3</sup>. We will show that some of the irregularities caused by verb form errors are consistent and can be taken into account.

*One goal of this paper is to recognize irregularities in parse trees caused by verb form errors, in order to increase recall.*

### 3.3 Overgeneralization

One potential consequence of allowing for irregularities in parse tree patterns is overgeneralization. For example, to allow for the “parse error” in §3.2 and to retrieve the word “*work*”, every determiner-less noun would potentially be turned into an *-ing* participle. This would clearly result in many invalid corrections. We propose using *n*-gram counts as a filter to counter this kind of overgeneralization.

*A second goal is to show that n-gram counts can effectively serve as a filter, in order to increase precision.*

## 4 Previous Research

This section discusses previous research on processing verb form errors, and contrasts verb form errors with those of the other parts-of-speech.

### 4.1 Verb Forms

Detection and correction of grammatical errors, including verb forms, have been explored in various applications. Hand-crafted error production rules (or “mal-rules”), augmenting a context-free grammar, are designed for a writing tutor aimed at deaf students (Michaud et al., 2000). Similar strategies with parse trees are pursued in (Bender et al., 2004), and error templates are utilized in (Heidorn, 2000) for a word processor. Carefully hand-crafted rules, when used alone, tend to yield high precision; they

<sup>3</sup>According to a study on parsing ungrammatical sentences (Foster, 2007), subject-verb and determiner-noun agreement errors can lower the F-score of a state-of-the-art probabilistic parser by 1.4%, and context-sensitive spelling errors (not verbs specifically), by 6%.

may, however, be less equipped to detect verb form errors within a perfectly grammatical sentence, such as the example given in §3.2.

An approach combining a hand-crafted context-free grammar and stochastic probabilities is pursued in (Lee and Seneff, 2006), but it is designed for a restricted domain only. A maximum entropy model, using lexical and POS features, is trained in (Izumi et al., 2003) to recognize a variety of errors. It achieves 55% precision and 23% recall overall, on evaluation data that partially overlap with those of the present paper. Unfortunately, results on verb form errors are not reported separately, and comparison with our approach is therefore impossible.

### 4.2 Other Parts-of-speech

Automatic error detection has been performed on other parts-of-speech, e.g., articles (Knight and Chander, 1994) and prepositions (Chodorow et al., 2007). The research issues with these parts-of-speech, however, are quite distinct. Relative to verb forms, errors in these categories do not “disturb” the parse tree as much. The process of feature extraction is thus relatively simple.

## 5 Data

### 5.1 Development Data

To investigate irregularities in parse tree patterns (see §3.2), we utilized the AQUAINT Corpus of English News Text. After parsing the corpus (Collins, 1997), we artificially introduced verb form errors into these sentences, and observed the resulting “disturbances” to the parse trees.

For disambiguation with *n*-grams (see §3.3), we made use of the WEB 1T 5-GRAM corpus. Prepared by Google Inc., it contains English *n*-grams, up to 5-grams, with their observed frequency counts from a large number of web pages.

### 5.2 Evaluation Data

Two corpora were used for evaluation. They were selected to represent two different genres, and two different mother tongues.

**JLE** (Japanese Learners of English corpus) This corpus is based on interviews for the Standard Speaking Test, an English-language proficiency test conducted in Japan (Izumi et al.,

Input	Hypothesized Correction		
	None	Valid	Invalid
w/ errors	<i>false_neg</i>	<i>true_pos</i>	<i>inv_pos</i>
w/o errors	<i>true_neg</i>	<i>false_pos</i>	

Table 4: Possible outcomes of a hypothesized correction.

2003). For 167 of the transcribed interviews, totalling 15,637 sentences<sup>4</sup>, grammatical errors were annotated and their corrections provided. By retaining the verb form errors<sup>5</sup>, but correcting all other error types, we generated a test set in which 477 sentences (3.1%) contain subject-verb agreement errors, and 238 (1.5%) contain auxiliary agreement and complementation errors.

**HKUST** This corpus<sup>6</sup> of short essays was collected from students, all native Chinese speakers, at the Hong Kong University of Science and Technology. It contains a total of 2556 sentences. They tend to be longer and have more complex structures than their counterparts in the JLE. Corrections are not provided; however, part-of-speech tags are given for the original words, and for the intended (but unwritten) corrections. Implications on our evaluation procedure are discussed in §5.4.

### 5.3 Evaluation Metric

For each verb in the input sentence, a change in verb form may be hypothesized. There are five possible outcomes for this hypothesis, as enumerated in Table 4. To penalize “false alarms”, a strict definition is used for false positives — even when the hypothesized correction yields a good sentence, it is still considered a false positive so long as the original sentence is acceptable.

It can sometimes be difficult to determine which words should be considered verbs, as they are not

<sup>4</sup>Obtained by segmenting (Reynar and Ratnaparkhi, 1997) the interviewee turns, and discarding sentences with only one word. The HKUST corpus was processed likewise.

<sup>5</sup>Specifically, those tagged with the “v\_fm1”, “v\_fin” (covering auxiliary agreement and complementation) and “v\_agr” (subject-verb agreement) types; those with semantic errors (see §2.1), i.e. “v\_tns” (tense), are excluded.

<sup>6</sup>Provided by Prof. John Milton, personal communication.

clearly demarcated in our evaluation corpora. We will thus apply the outcomes in Table 4 at the sentence level; that is, the output sentence is considered a true positive only if the original sentence contains errors, and only if valid corrections are offered for all errors.

The following statistics are computed:

**Accuracy** The proportion of sentences which, after being treated by the system, have correct verb forms. That is,  $(true\_neg + true\_pos)$  divided by the total number of sentences.

**Recall** Out of all sentences with verb form errors, the percentage whose errors have been successfully corrected by the system. That is,  $true\_pos$  divided by  $(true\_pos + false\_neg + inv\_pos)$ .

**Detection Precision** This is the first of two types of precision to be reported, and is defined as follows: Out of all sentences for which the system has hypothesized corrections, the percentage that actually contain errors, without regard to the validity of the corrections. That is,  $(true\_pos + inv\_pos)$  divided by  $(true\_pos + inv\_pos + false\_pos)$ .

**Correction Precision** This is the more stringent type of precision. In addition to successfully determining that a correction is needed, the system must offer a valid correction. Formally, it is  $true\_pos$  divided by  $(true\_pos + false\_pos + inv\_pos)$ .

### 5.4 Evaluation Procedure

For the JLE corpus, all figures above will be reported. The HKUST corpus, however, will not be evaluated on subject-verb agreement, since a sizable number of these errors are induced by other changes in the sentence<sup>7</sup>.

Furthermore, the HKUST corpus will require manual evaluation, since the corrections are not annotated. Two native speakers of English were given the edited sentences, as well as the original input. For each pair, they were asked to select one of four statements: one of the two is better, or both are equally correct, or both are equally incorrect. The

<sup>7</sup>e.g., the subject of the verb needs to be changed from singular to plural.

Expected Tree {⟨usage⟩,...}	Tree disturbed by substitution [⟨crr⟩ → ⟨err⟩]
{ING <sub>prog</sub> ,ED <sub>pass</sub> }	<i>A dog is [sleeping→sleep]. I'm [living→live] in XXX city.</i>
<pre> graph TD   VP1[VP] --- be1[be]   VP1 --- VP2[VP]   VP2 --- crr1["crr/{VBG, VBN}"] </pre>	<pre> graph TD   VP3[VP] --- be3[be]   VP3 --- NP3[NP]   NP3 --- err3["err/NN"]   VP4[VP] --- be4[be]   VP4 --- ADJP4[ADJP]   ADJP4 --- err4["err/JJ"] </pre>
{ING <sub>verb</sub> ,INF <sub>verb</sub> }	<i>I like [skiing→ski] very much; She likes to [go→going] around</i>
<pre> graph TD   VP5[VP] --- V5["*/V"]   VP5 --- SG5[SG]   SG5 --- VP6[VP]   VP6 --- crr5["crr/{VBG, TO} ..."] </pre>	<pre> graph TD   VP7[VP] --- V7["*/V"]   VP7 --- NP7[NP]   NP7 --- err7["err/NN"]   VP8[VP] --- V8["*/V"]   VP8 --- PP8[PP]   PP8 --- to8["to/TO"]   PP8 --- SG8[SG]   SG8 --- VP9[VP]   VP9 --- err9["err/VBG"] </pre>
ING <sub>prep</sub>	<i>I lived in France for [studying→study] French language.</i>
<pre> graph TD   PP10[PP] --- IN10["*/IN"]   PP10 --- SG10[SG]   SG10 --- VP11[VP]   VP11 --- crr10["crr/VBG ..."] </pre>	<pre> graph TD   PP12[PP] --- IN12["*/IN"]   PP12 --- NP12[NP]   NP12 --- err12["err/NN"] </pre>

Table 5: Effects of incorrect verb forms on parse trees. The left column shows trees normally expected for the indicated usages (see Table 3). The right column shows the resulting trees when the correct verb form ⟨crr⟩ is replaced by ⟨err⟩. Detailed comments are provided in §6.1.

correction precision is thus the proportion of pairs where the edited sentence is deemed better. Accuracy and recall cannot be computed, since it was impossible to distinguish syntactic errors from semantic ones (see §2).

### 5.5 Baselines

Since the vast majority of verbs are in their correct forms, the *majority baseline* is to propose no correction. Although trivial, it is a surprisingly strong baseline, achieving more than 98% for auxiliary agreement and complementation in JLE, and just shy of 97% for subject-verb agreement.

For auxiliary agreement and complementation, the *verb-only baseline* is also reported. It attempts corrections only when the word in question is actu-

ally tagged as a verb. That is, it ignores the spurious noun- and adjectival phrases in the parse tree discussed in §3.2, and relies only on the output of the part-of-speech tagger.

## 6 Experiments

Corresponding to the issues discussed in §3.2 and §3.3, our experiment consists of two main steps.

### 6.1 Derivation of Tree Patterns

Based on (Quirk et al., 1985), we observed tree patterns for a set of verb form usages, as summarized in Table 3. Using these patterns, we introduced verb form errors into AQUAINT, then re-parsed the corpus (Collins, 1997), and compiled the changes in the “disturbed” trees into a catalog.

<i>N</i> -gram	Example
be {ING <sub>prog</sub> , ED <sub>pass</sub> } *	The dog <i>is sleeping</i> . The door <i>is open</i> .
verb {ING <sub>verb</sub> , INF <sub>verb</sub> } *	I need <i>to do</i> this. I need <i>beef</i> for the curry.
verb <sub>1</sub> *ing and {ING <sub>verb</sub> , INF <sub>verb</sub> }	<i>enjoy</i> reading and <i>going</i> to pachinko <i>go</i> shopping and <i>have</i> dinner
prep {ING <sub>prep</sub> } *	for <i>studying</i> French language a class for <i>sign</i> language
have {ED <sub>perf</sub> } *	I have <i>rented</i> a video I have <i>lunch</i> in Ginza

Table 6: The *n*-grams used for filtering, with examples of sentences which they are intended to differentiate. The hypothesized usages (shown in the curly brackets) as well as the original verb form, are considered. For example, the first sentence is originally “*The dog is \*sleep.*” The three trigrams “*is sleeping .*”, “*is slept .*” and “*is sleep .*” are compared; the first trigram has the highest count, and the correction “*sleeping*” is therefore applied.

A portion of this catalog<sup>8</sup> is shown in Table 5. Comments on {ING<sub>prog</sub>, ED<sub>pass</sub>} can be found in §3.2. Two cases are shown for {ING<sub>verb</sub>, INF<sub>verb</sub>}. In the first case, an *-ing* participle in verb complementation is reduced to its base form, resulting in a noun phrase. In the second, an infinitive is constructed with the *-ing* participle rather than the base form, causing “*to*” to be misconstrued as a preposition. Finally, in ING<sub>prep</sub>, an *-ing* participle in preposition complementation is reduced to its base form, and is subsumed in a noun phrase.

## 6.2 Disambiguation with N-grams

The tree patterns derived from the previous step may be considered as the “necessary” conditions for proposing a change in verb forms. They are not “sufficient”, however, since they tend to be overly general. Indiscriminate application of these patterns on AQUAINT would result in false positives for 46.4% of the sentences.

For those categories with a high rate of false positives (all except BASE<sub>md</sub>, BASE<sub>do</sub> and FINITE), we utilized *n*-grams as filters, allowing a correction only when its *n*-gram count in the WEB 1T 5-GRAM

<sup>8</sup>Due to space constraints, only those trees with significant changes above the leaf level are shown.

Hyp. Usage	False Pos.	Hypothesized Usage	False Pos.
BASE <sub>md</sub>	16.2%	{ING <sub>verb</sub> , INF <sub>verb</sub> }	33.9%
BASE <sub>do</sub>	0.9%	{ING <sub>prog</sub> , ED <sub>pass</sub> }	21.0%
FINITE	12.8%	ING <sub>prep</sub>	13.7%
		ED <sub>perf</sub>	1.4%

Table 7: The distribution of false positives in AQUAINT. The total number of false positives is 994, represents less than 1% of the 100,000 sentences drawn from the corpus.

corpus is greater than that of the original. The filtering step reduced false positives from 46.4% to less than 1%. Table 6 shows the *n*-grams, and Table 7 provides a breakdown of false positives in AQUAINT after *n*-gram filtering.

## 6.3 Results for Subject-Verb Agreement

In JLE, the accuracy of subject-verb agreement error correction is 98.93%. Compared to the majority baseline of 96.95%, the improvement is statistically significant<sup>9</sup>. Recall is 80.92%; detection precision is 83.93%, and correction precision is 81.61%.

Most mistakes are caused by misidentified subjects. Some *wh*-questions prove to be especially difficult, perhaps due to their relative infrequency in newswire texts, on which the parser is trained. One example is the question “*How much extra time does the local train \*takes?*”. The word “*does*” is not recognized as a “do”-support, and so the verb “*take*” was mistakenly turned into a third person form to agree with “*train*”.

## 6.4 Results for Auxiliary Agreement & Complementation

Table 8 summarizes the results for auxiliary agreement and complementation, and Table 2 shows some examples of real sentences corrected by the system. Our proposed method yields 98.94% accuracy. It is a statistically significant improvement over the majority baseline (98.47%), although not significant over the verb-only baseline<sup>10</sup> (98.85%), perhaps a reflection of the small number of test sentences with verb form errors. The Kappa statistic for the man-

<sup>9</sup> $p < 0.005$  according to McNemar’s test.

<sup>10</sup>With  $p = 1 * 10^{-10}$  and  $p = 0.038$ , respectively, according to McNemar’s test



Corpus	Method	Accuracy	Precision (correction)	Precision (detection)	Recall
JLE	verb-only	98.85%	71.43%	84.75%	31.51%
	all	98.94%	68.00%	80.67%	42.86%
HKUST	all	not available	71.71%	not available	

Table 8: Results on the JLE and HKUST corpora for auxiliary agreement and complementation. The majority baseline accuracy is 98.47% for JLE. The verb-only baseline accuracy is 98.85%, as indicated on the second row. “All” denotes the complete proposed method. See §6.4 for detailed comments.

Usage	JLE	HKUST
	Count (Prec.)	Count (Prec.)
BASE <sub>md</sub>	13 (92.3%)	25 (80.0%)
BASE <sub>do</sub>	5 (100%)	0
FINITE	9 (55.6%)	0
ED <sub>perf</sub>	11 (90.9%)	3 (66.7%)
{ING <sub>prog</sub> , ED <sub>pass</sub> }	54 (58.6%)	30 (70.0%)
{ING <sub>verb</sub> , INF <sub>verb</sub> }	45 (60.0%)	16 (59.4%)
ING <sub>prep</sub>	10 (60.0%)	2 (100%)

Table 9: Correction precision of individual correction patterns (see Table 5) on the JLE and HKUST corpus.

ual evaluation of HKUST is 0.76, corresponding to “substantial agreement” between the two evaluators (Landis and Koch, 1977). The correction precisions for the JLE and HKUST corpora are comparable.

Our analysis will focus on {ING<sub>prog</sub>, ED<sub>pass</sub>} and {ING<sub>verb</sub>, INF<sub>verb</sub>}, two categories with relatively numerous correction attempts and low precisions, as shown in Table 9. For {ING<sub>prog</sub>, ED<sub>pass</sub>}, many invalid corrections are due to wrong predictions of voice, which involve semantic choices (see §2.1). For example, the sentence “... the main duty is study well” is edited to “... the main duty is studied well”, a grammatical sentence but semantically unlikely.

For {ING<sub>verb</sub>, INF<sub>verb</sub>}, a substantial portion of the false positives are valid, but unnecessary, corrections. For example, there is no need to turn “I like cooking” into “I like to cook”, as the original is perfectly acceptable. Some kind of confidence measure on the  $n$ -gram counts might be appropriate for reducing such false alarms.

Characteristics of speech transcripts pose some further problems. First, colloquial expressions, such as the word “like”, can be tricky to process. In the

question “Can you like give me the money back”, “like” is misconstrued to be the main verb, and “give” is turned into an infinitive, resulting in “Can you like \*to give me the money back”. Second, there are quite a few incomplete sentences that lack subjects for the verbs. No correction is attempted on them.

Also left uncorrected are misused forms in non-finite clauses that describe a noun. These are typically base forms that should be replaced with *-ing* participles, as in “The girl \*wear a purple skiwear is a student of this ski school”. Efforts to detect this kind of error had resulted in a large number of false alarms.

Recall is further affected by cases where a verb is separated from its auxiliary or main verb by many words, often with conjunctions and other verbs in between. One example is the sentence “I used to climb up the orange trees and \*catching insects”. The word “catching” should be an infinitive complementing “used”, but is placed within a noun phrase together with “trees” and “insects”.

## 7 Conclusion

We have presented a method for correcting verb form errors. We investigated the ways in which verb form errors affect parse trees. When allowed for, these unusual tree patterns can expand correction coverage, but also tend to result in overgeneration of hypothesized corrections.  $N$ -grams have been shown to be an effective filter for this problem.

## 8 Acknowledgments

We thank Prof. John Milton for the HKUST corpus, Tom Lee and Ken Schutte for their assistance with the evaluation, and the anonymous reviewers for their helpful feedback.

## References

- E. Bender, D. Flickinger, S. Oepen, A. Walsh, and T. Baldwin. 2004. Arboretum: Using a Precision Grammar for Grammar Checking in CALL. *Proc. INSTIL/ICALL Symposium on Computer Assisted Learning*.
- M. Chodorow, J. R. Tetreault, and N.-R. Han. 2007. Detection of Grammatical Errors Involving Prepositions. In *Proc. ACL-SIGSEM Workshop on Prepositions*. Prague, Czech Republic.
- M. Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. *Proc. ACL*.
- J. Foster. 2007. Treebanks Gone Bad: Generating a Treebank of Ungrammatical English. In *Proc. IJCAI Workshop on Analytics for Noisy Unstructured Data*. Hyderabad, India.
- G. Heidorn. 2000. Intelligent Writing Assistance. *Handbook of Natural Language Processing*. Robert Dale, Hermann Moisi and Harold Somers (ed.). Marcel Dekker, Inc.
- E. Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic Error Detection in the Japanese Learner's English Spoken Data. In *Companion Volume to Proc. ACL*. Sapporo, Japan.
- K. Knight and I. Chander. 1994. Automated Postediting of Documents. In *Proc. AAAI*. Seattle, WA.
- J. R. Landis and G. G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33(1):159–174.
- L. Michaud, K. McCoy and C. Pennington. 2000. An Intelligent Tutoring System for Deaf Learners of Written English. *Proc. 4th International ACM Conference on Assistive Technologies*.
- J. Lee and S. Seneff. 2006. Automatic Grammar Correction for Second-Language Learners. In *Proc. Interspeech*. Pittsburgh, PA.
- J. C. Reynar and A. Ratnaparkhi. 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proc. 5th Conference on Applied Natural Language Processing*. Washington, D.C.
- R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, New York.

# Hypertagging: Supertagging for Surface Realization with CCG

Dominic Espinosa and Michael White and Dennis Mehay

Department of Linguistics

The Ohio State University

Columbus, OH, USA

{espinosa,mwhite,mehay}@ling.osu.edu

## Abstract

In lexicalized grammatical formalisms, it is possible to separate *lexical category assignment* from the combinatory processes that make use of such categories, such as parsing and realization. We adapt techniques from *supertagging* — a relatively recent technique that performs complex lexical tagging before full parsing (Bangalore and Joshi, 1999; Clark, 2002) — for chart realization in OpenCCG, an open-source NLP toolkit for CCG. We call this approach *hypertagging*, as it operates at a level “above” the syntax, tagging semantic representations with syntactic lexical categories. Our results demonstrate that a hypertagger-informed chart realizer can achieve substantial improvements in realization speed (being approximately twice as fast) with superior realization quality.

## 1 Introduction

In lexicalized grammatical formalisms such as Lexicalized Tree Adjoining Grammar (Schabes et al., 1988, LTAG), Combinatory Categorical Grammar (Steedman, 2000, CCG) and Head-Driven Phrase-Structure Grammar (Pollard and Sag, 1994, HPSG), it is possible to separate *lexical category assignment* — the assignment of informative syntactic categories to linguistic objects such as words or lexical predicates — from the combinatory processes that make use of such categories — such as parsing and surface realization. One way of performing lexical assignment is simply to hypothesize all possible lexical categories and then search for the best

combination thereof, as in the CCG parser in (Hockenmaier, 2003) or the chart realizer in (Carroll and Oepen, 2005). A relatively recent technique for lexical category assignment is *supertagging* (Bangalore and Joshi, 1999), a preprocessing step to parsing that assigns likely categories based on word and part-of-speech (POS) contextual information. Supertagging was dubbed “almost parsing” by these authors, because an oracle supertagger left relatively little work for their parser, while speeding up parse times considerably. Supertagging has been more recently extended to a multitagging paradigm in CCG (Clark, 2002; Curran et al., 2006), leading to extremely efficient parsing with state-of-the-art dependency recovery (Clark and Curran, 2007).

We have adapted this multitagging approach to lexical category assignment for realization using the CCG-based natural language toolkit OpenCCG.<sup>1</sup> Instead of basing category assignment on linear word and POS context, however, we predict lexical categories based on contexts within a directed graph structure representing the logical form (LF) of a proposition to be realized. Assigned categories are instantiated in OpenCCG’s chart realizer where, together with a treebank-derived syntactic grammar (Hockenmaier and Steedman, 2007) and a factored language model (Bilmes and Kirchhoff, 2003), they constrain the English word-strings that are chosen to express the LF. We have dubbed this approach *hypertagging*, as it operates at a level “above” the syntax, moving from semantic representations to syntactic categories.

We evaluate this hypertagger in two ways: first,

<sup>1</sup><http://openccg.sourceforge.net>.

we evaluate it as a tagger, where the hypertagger achieves high single-best (93.6%) and multitagging labelling accuracies (95.8–99.4% with category per lexical predication ratios ranging from 1.1 to 3.9).<sup>2</sup> Second, we compare a hypertagger-augmented version of OpenCCG’s chart realizer with the pre-existing chart realizer (White et al., 2007) that simply instantiates the chart with all possible CCG categories (subject to frequency cutoffs) for each input LF predicate. The hypertagger-seeded realizer runs approximately twice as fast as the pre-existing OpenCCG realizer and finds a larger number of complete realizations, resorting less to chart fragment assembly in order to produce an output within a 15 second per-sentence time limit. Moreover, the overall BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007) scores, as well as numbers of exact string matches (as measured against to the original sentences in the CCGbank) are higher for the hypertagger-seeded realizer than for the pre-existing realizer.

This paper is structured as follows: Section 2 provides background on chart realization in OpenCCG using a corpus-derived grammar. Section 3 describes our hypertagging approach and how it is integrated into the realizer. Section 4 describes our results, followed by related work in Section 5 and our conclusions in Section 6.

## 2 Background

### 2.1 Surface Realization with OpenCCG

The OpenCCG surface realizer is based on Steedman’s (2000) version of CCG elaborated with Baldrige and Kruijff’s multi-modal extensions for lexically specified derivation control (Baldrige, 2002; Baldrige and Kruijff, 2003) and hybrid logic dependency semantics (Baldrige and Kruijff, 2002). OpenCCG implements a symbolic-statistical chart realization algorithm (Kay, 1996; Carroll et al., 1999; White, 2006b) combining (1) a theoretically grounded approach to syntax and semantic composition with (2) factored language models (Bilmes and Kirchhoff, 2003) for making choices among the options left open by the grammar.

In OpenCCG, the search for complete realizations

<sup>2</sup>Note that the multitagger is “correct” if the correct tag is anywhere in the multitag set.

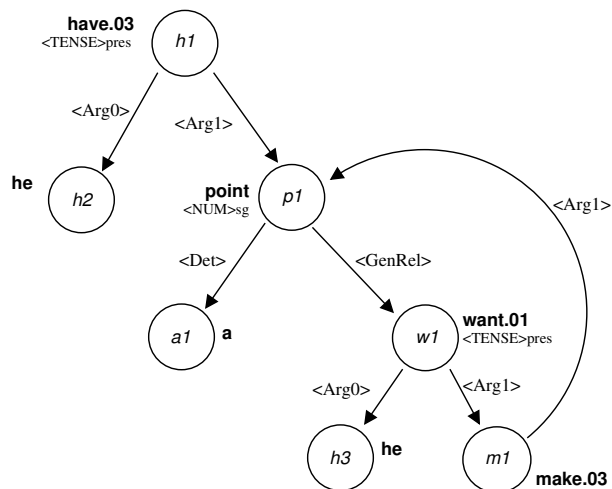


Figure 1: Semantic dependency graph from the CCGbank for *He has a point he wants to make [...]*

makes use of  $n$ -gram language models over words represented as vectors of factors, including surface form, part of speech, supertag and semantic class. The search proceeds in one of two modes, *anytime* or *two-stage* (packing/unpacking). In the anytime mode, a best-first search is performed with a configurable time limit: the scores assigned by the  $n$ -gram model determine the order of the edges on the agenda, and thus have an impact on realization speed. In the two-stage mode, a packed forest of all possible realizations is created in the first stage; in the second stage, the packed representation is unpacked in bottom-up fashion, with scores assigned to the edge for each sign as it is unpacked, much as in (Langkilde, 2000). Edges are grouped into equivalence classes when they have the same syntactic category and cover the same parts of the input logical form. Pruning takes place within equivalence classes of edges. Additionally, to realize a wide range of paraphrases, OpenCCG implements an algorithm for efficiently generating from disjunctive logical forms (White, 2006a).

To illustrate the input to OpenCCG, consider the semantic dependency graph in Figure 1, which is taken from section 00 of a Propbank-enhanced version of the CCGbank (Boxwell and White, 2008). In the graph, each node has a lexical predication (e.g. **make.03**) and a set of semantic features (e.g.  $\langle \text{NUM} \rangle \text{sg}$ ); nodes are connected via dependency relations (e.g.  $\langle \text{ARG0} \rangle$ ). Internally, such

graphs are represented using Hybrid Logic Dependency Semantics (HLDS), a dependency-based approach to representing linguistic meaning developed by Baldridge and Kruijff (2002). In HLDS, hybrid logic (Blackburn, 2000) terms are used to describe dependency graphs. These graphs have been suggested as representations for discourse structure, and have their own underlying semantics (White, 2006b).

To more robustly support broad coverage surface realization, OpenCCG has recently been enhanced to greedily assemble fragments in the event that the realizer fails to find a complete realization. The fragment assembly algorithm begins with the edge for the best partial realization, i.e. the one that covers the most elementary predications in the input logical form, with ties broken according to the n-gram score. (Larger fragments are preferred under the assumption that they are more likely to be grammatical.) Next, the chart and agenda are greedily searched for the best edge whose semantic coverage is disjoint from those selected so far; this process repeats until no further edges can be added to the set of selected fragments. In the final step, these fragments are concatenated, again in a greedy fashion, this time according to the n-gram score of the concatenated edges: starting with the original best edge, the fragment whose concatenation on the left or right side yields the highest score is chosen as the one to concatenate next, until all the fragments have been concatenated into a single output.

## 2.2 Realization from an Enhanced CCGbank

White et al. (2007) describe an ongoing effort to engineer a grammar from the CCGbank (Hockenmaier and Steedman, 2007) — a corpus of CCG derivations derived from the Penn Treebank — suitable for realization with OpenCCG. This process involves converting the corpus to reflect more precise analyses, where feasible, and adding semantic representations to the lexical categories. In the first step, the derivations in the CCGbank are revised to reflect the desired syntactic derivations. Changes to the derivations are necessary to reflect the lexicalized treatment of coordination and punctuation assumed by the multi-modal version of CCG that is implemented in OpenCCG. Further changes are necessary to support semantic dependencies rather than surface syn-

tactic ones; in particular, the features and unification constraints in the categories related to semantically empty function words such complementizers, infinitival-*to*, expletive subjects, and case-marking prepositions are adjusted to reflect their purely syntactic status.

In the second step, a grammar is extracted from the converted CCGbank and augmented with logical forms. Categories and unary type changing rules (corresponding to zero morphemes) are sorted by frequency and extracted if they meet the specified frequency thresholds.

A separate transformation then uses around two dozen generalized templates to add logical forms to the categories, in a fashion reminiscent of (Bos, 2005). The effect of this transformation is illustrated below. Example (1) shows how numbered semantic roles, taken from PropBank (Palmer et al., 2005) when available, are added to the category of an active voice, past tense transitive verb, where **\*pred\*** is a placeholder for the lexical predicate; examples (2) and (3) show how more specific relations are introduced in the category for determiners and the category for the possessive 's, respectively.

- (1)  $s_{1:dcl} \setminus np_2 / np_3 \implies$   
 $s_{1:dcl,x1} \setminus np_{2:x2} / np_{3:x3} : @_{x1}(*pred* \wedge$   
 $\langle TENSE \rangle pres \wedge \langle ARG0 \rangle x2 \wedge \langle ARG1 \rangle x3)$
- (2)  $np_1 / n_1 \implies$   
 $np_{1:x1} / n_{1:x1} : @_{x1}(\langle DET \rangle (d \wedge *pred*))$
- (3)  $np_1 / n_1 \setminus np_2 \implies$   
 $np_{1:x1} / n_{1:x1} \setminus np_{2:x2} : @_{x1}(\langle GENOWN \rangle x2)$

After logical form insertion, the extracted and augmented grammar is loaded and used to parse the sentences in the CCGbank according to the gold-standard derivation. If the derivation can be successfully followed, the parse yields a logical form which is saved along with the corpus sentence in order to later test the realizer. The algorithm for following corpus derivations attempts to continue processing if it encounters a blocked derivation due to sentence-internal punctuation. While punctuation has been partially reanalyzed to use lexical categories, many problem cases remain due to the CCGbank's reliance on punctuation-specific binary rules that are not supported in OpenCCG.

Currently, the algorithm succeeds in creating logical forms for 97.7% of the sentences in the development section (Sect. 00) of the converted CCGbank, and 96.1% of the sentences in the test section (Sect. 23). Of these, 76.6% of the development logical forms are semantic dependency graphs with a single root, while 76.7% of the test logical forms have a single root. The remaining cases, with multiple roots, are missing one or more dependencies required to form a fully connected graph. These missing dependencies usually reflect inadequacies in the current logical form templates.

### 2.3 Factored Language Models

Following White et al. (2007), we use factored trigram models over words, part-of-speech tags and supertags to score partial and complete realizations. The language models were created using the SRILM toolkit (Stolcke, 2002) on the standard training sections (2–21) of the CCGbank, with sentence-initial words (other than proper names) uncapitalized. While these models are considerably smaller than the ones used in (Langkilde-Geary, 2002; Veldal and Oepen, 2005), the training data does have the advantage of being in the same domain and genre (using larger n-gram models remains for future investigation). The models employ interpolated Kneser-Ney smoothing with the default frequency cutoffs. The best performing model interpolates a word trigram model with a trigram model that chains a POS model with a supertag model, where the POS model conditions on the previous two POS tags, and the supertag model conditions on the previous two POS tags as well as the current one.

Note that the use of supertags in the factored language model to score possible realizations is distinct from the prediction of supertags for lexical category assignment: the former takes the words in the local context into account (as in supertagging for parsing), while the latter takes features of the logical form into account. It is this latter process which we call hyper-tagging, and to which we now turn.

## 3 The Approach

### 3.1 Lexical Smoothing and Search Errors

In White et al.’s (2007) initial investigation of scaling up OpenCCG for broad coverage realization,

test set	grammar	complete oracle / best
dev (00)	dev	49.1% / 47.8%
	train	37.5% / 22.6%

Table 1: Percentage of complete realizations using an oracle n-gram model versus the best performing factored language model.

all categories observed more often than a threshold frequency were instantiated for lexical predicates; for unseen words, a simple smoothing strategy based on the part of speech was employed, assigning the most frequent categories for the POS. This approach turned out to suffer from a large number of search errors, where the realizer failed to find a complete realization before timing out even in cases where the grammar supported one. To confirm that search errors had become a significant issue, White et al. compared the percentage of complete realizations (versus fragmentary ones) with their top scoring model against an oracle model that uses a simplified BLEU score based on the target string, which is useful for regression testing as it guides the best-first search to the reference sentence. The comparison involved both a medium-sized (non-blind) grammar derived from the development section and a large grammar derived from the training sections (the latter with slightly higher thresholds). As shown in Table 1, with the large grammar derived from the training sections, many fewer complete realizations are found (before timing out) using the factored language model than are possible, as indicated by the results of using the oracle model. By contrast, the difference is small with the medium-sized grammar derived from the development section. This result is not surprising when one considers that a large number of common words are observed to have many possible categories.

In the next section, we show that a supertagger for CCG realization, or hypertagger, can reduce the problem of search errors by focusing the search space on the most likely lexical categories.

### 3.2 Maximum Entropy Hypertagging

As supertagging for parsing involves studying a given input word and its local context, the concep-

tual equivalent for a lexical predicate in the LF is to study a given node and its local graph structure. Our implementation makes use of three general types of features: lexicalized features, which are simply the names of the parent and child elementary predication nodes, graph structural features, such as the total number of edges emanating from a node, the number of argument and non-argument dependents, and the names of the relations of the dependent nodes to the parent node, and syntactico-semantic attributes of nodes, such as the tense and number. For example, in the HLDS graph shown in Figure 1, the node representing *want* has two dependents, and the relational type of *make* with respect to *want* is ARG1.

Clark (2002) notes in his parsing experiments that the POS tags of the surrounding words are highly informative. As discussed below, a significant gain in hypertagging accuracy resulted from including features sensitive to the POS tags of a node’s parent, the node itself, and all of its arguments and modifiers. Predicting these tags requires the use of a separate POS tagger, which operates in a manner similar to the hypertagger itself, though exploiting a slightly different set of features (e.g., including features corresponding to the four-character prefixes and suffixes of rare logical predication names). Following the (word) supertagging experiments of (Curran et al., 2006) we assigned potentially multiple POS tags to each elementary predication. The POS tags assigned are all those that are some factor  $\beta$  of the highest ranked tag,<sup>3</sup> giving an average of 1.1 POS tags per elementary predication. The values of the corresponding feature functions are the POS tag probabilities according to the POS tagger. At this ambiguity level, the POS tagger is correct  $\approx 92\%$  of the time.

Features for the hypertagger were extracted from semantic dependency graphs extracted from sections 2 through 21 of the CCGbank. In total, 37,168 dependency graphs were derived from the corpus, yielding 468,628 feature parameters.

The resulting contextual features and gold-standard supertag for each predication were then used to train a maximum entropy classifier model.

<sup>3</sup>I.e., all tags  $t$  whose probabilities  $p(t) \geq \beta \cdot p^*$ , where  $p^*$  is the highest ranked tag’s probability.

Maximum entropy models describe a set of probability distributions of the form:

$$p(o | x) = \frac{1}{Z(x)} \cdot \exp\left(\sum_{i=1}^n \lambda_i f_i(o, x)\right)$$

where  $o$  is an outcome,  $x$  is a context, the  $f_i$  are feature functions, the  $\lambda_i$  are the respective weights of the feature functions, and  $Z(x)$  is a normalizing sum over all competing outcomes. More concretely, given an elementary predication labeled *want* (as in Figure 1), a feature function over this node could be:

$$f(o, x) = \begin{cases} 1, & \text{if } o \text{ is } (s[\text{dcl}]\backslash\text{np})/(s[\text{adj}]\backslash\text{np}) \text{ and} \\ & \text{number\_of\_LF\_dependents}(x) = 2 \\ 0, & \text{otherwise.} \end{cases}$$

We used Zhang Le’s maximum entropy toolkit<sup>4</sup> for training the hypertagging model, which uses an implementation of Limited-memory BFGS, an approximate quasi-Newton optimization method from the numerical optimization literature (Liu and Nocedal, 1989). Using L-BFGS allowed us to include continuous feature function values where appropriate (e.g., the probabilities of automatically-assigned POS tags). We trained each hypertagging model to 275 iterations and our POS tagging model to 400 iterations. We used no feature frequency cut-offs, but rather employed Gaussian priors with global variances of 100 and 75, respectively, for the hypertagging and POS tagging models.

### 3.3 Iterative $\beta$ -Best Realization

During realization, the hypertagger serves to probabilistically filter the categories assigned to an elementary predication, as well as to propose categories for rare or unseen predicates. Given a predication, the tagger returns a  $\beta$ -best list of supertags in order of decreasing probability. Increasing the number of categories returned clearly increases the likelihood that the most-correct supertag is among them, but at a corresponding cost in chart size. Accordingly, the hypertagger begins with a highly restrictive value for  $\beta$ , and backs off to progressively less-restrictive values if no complete realization could be found using the set of supertags returned. The search is restarted

<sup>4</sup>[http://homepages.inf.ed.ac.uk/s0450736/maxent\\_toolkit.html](http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html).

Table 2: Hypertagger accuracy on Sections 00 and 23. Results (in percentages) are for per-logical-predication (PR) and per-whole-graph (GRPH) tagging accuracies. Difference between best-only and baselines (b.l.) is significant ( $p < 2 \cdot 10^{-16}$ ) by McNemar’s  $\chi^2$  test.

$\beta$	Tags Pred	Sect00		Sect23	
		PR	GRPH	PR	GRPH
b.l. 1	1	68.7	1.8	68.7	2.3
b.l. 2	2	84.3	9.9	84.4	10.9
1.0	1	93.6	40.4	93.6	38.2
0.16	1.1	95.8	55.7	96.2	56.8
0.05	1.2	96.6	63.8	97.3	66.0
0.0058	1.5	97.9	74.8	98.3	76.9
1.75e-3	1.8	98.4	78.9	98.7	81.8
6.25e-4	2.2	98.7	82.5	99.0	84.3
1.25e-4	3.2	99.0	85.7	99.3	88.5
5.8e-5	3.9	99.1	87.2	99.4	89.9

from scratch with the next  $\beta$  value, though in principle the same chart could be expanded. The iterative,  $\beta$ -best search for a complete realization uses the realizer’s packing mode, which can more quickly determine whether a complete realization is possible. If the halfway point of the overall time limit is reached with no complete realization, the search switches to best-first mode, ultimately assembling fragments if no complete realization can be found during the remaining time.

## 4 Results and Discussion

Several experiments were performed in training and applying the hypertagger. Three different models were created using 1) non-lexicalized features only, 2) all features excluding POS tags, 3) all, 3) all features except syntactico-semantic attributes such as tense and number and 4) all features available. Models trained on these feature subsets were tested against one another on Section 00, and then the best performing model was run on both Section 00 and 23.

### 4.1 Feature Ablation Testing

The the whole feature set was found in feature ablation testing on the development set to outperform all other feature subsets significantly ( $p < 2.2 \cdot 10^{-16}$ ). These results listed in Table 3. As we can see, taking

Table 3: Hypertagger feature ablation testing results on Section 00. The full feature set outperforms all others significantly ( $p < 2.2 \cdot 10^{-16}$ ). Results for per-predication (PR) and per-whole-graph (GRPH) tagging percentage accuracies are listed. (**Key:** **no-POS**=no POS features; **no-attr**=no syntactico-semantic attributes such as tense and number; **non-lex**=non-lexicalized features only (no predication names).

FEATURESET	PR	GRPH
full	93.6	40.37
no-POS	91.3	29.5
no-attr	91.8	31.2
non-lex	91.5	28.7

away any one class of features leads to drop in per-predication tagging accuracy of at least 1.8% and a drop per-whole-graph accuracy of at least 9.2%. As expected from previous work in supertagging (for parsing), POS features resulted in a large improvement in overall accuracy (1.8%). Although the POS tagger by itself is only 92% accurate (as a multi-tagger of 1.1  $\frac{POS}{word}$  average ambiguity) — well below the state-of-the-art for the tagging of words — its predictions are still quite valuable to the hypertagger.

### 4.2 Best Model Hypertagger Accuracy

The results for the full feature set on Sections 00 and 23 are outlined in Table 2. Included in this table are accuracy data for a baseline dummy tagger which simply assigns the most-frequently-seen tag(s) for a given predication and backs off to the overall most frequent tag(s) when confronted with an unseen predication. The development set (00) was used to tune the  $\beta$  parameter to obtain reasonable hypertag ambiguity levels; the model was not otherwise tuned to it. The hypertagger achieves high per-predication and whole-graph accuracies even at small ambiguity levels.

### 4.3 Realizer Performance

Tables 4 and 5 show how the hypertagger improves realization performance on the development and test sections of the CCGbank. As Table 4 indicates, using the hypertagger in an iterative beta-best fashion more than doubles the number of grammatically complete realizations found within the time



Table 5: Realization quality metrics exact match, BLEU and METEOR, on complete realizations only and overall, with and without hypertagger, on Sections 00 and 23.

Section	Hypertagger	Complete		Overall		
		BLEU	METEOR	Exact	BLEU	METEOR
00	with	0.8137	0.9153	15.3%	0.6567	0.8494
	w/o	0.6864	0.8585	11.3%	0.5902	0.8209
23	with	<b>0.8149</b>	0.9162	16.0%	<b>0.6701</b>	0.8557
	w/o	0.6910	0.8606	12.3%	0.6022	0.8273

Table 4: Percentage of grammatically complete realizations, runtimes for complete realizations and overall runtimes, with and without hypertagger, on Sections 00 and 23.

Section	Hypertagger	Percent Complete	Complete Time	Overall Time
00	with	47.4%	1.2s	4.5s
	w/o	22.6%	8.7s	9.5s
23	with	48.5%	<b>1.2s</b>	<b>4.4s</b>
	w/o	23.5%	8.9s	9.6s

limit; on the development set, this improvement eliminates more than the number of known search errors (cf. Table 1). Additionally, by reducing the search space, the hypertagger cuts overall realization times by more than half, and in the cases where complete realizations are found, realization times are reduced by a factor of four, down to 1.2 seconds per sentence on a desktop Linux PC.

Table 5 shows that increasing the number of complete realizations also yields improved BLEU and METEOR scores, as well as more exact matches. In particular, the hypertagger makes possible a more than 6-point improvement in the overall BLEU score on both the development and test sections, and a more than 12-point improvement on the sentences with complete realizations.

As the effort to engineer a grammar suitable for realization from the CCGbank proceeds in parallel to our work on hypertagging, we expect the hypertagger-seeded realizer to continue to improve, since a more complete and precise extracted grammar should enable more complete realizations to be found, and richer semantic representations should

simplify the hypertagging task. Even with the current incomplete set of semantic templates, the hypertagger brings realizer performance roughly up to state-of-the-art levels, as our overall test set BLEU score (0.6701) slightly exceeds that of Cahill and van Genabith (2006), though at a coverage of 96% instead of 98%. We caution, however, that it remains unclear how meaningful it is to directly compare these scores when the realizer inputs vary considerably in their specificity, as Langkilde-Geary’s (2002) experiments dramatically illustrate.

## 5 Related Work

Our approach follows Langkilde-Geary (2002) and Callaway (2003) in aiming to leverage the Penn Treebank to develop a broad-coverage surface realizer for English. However, while these earlier, generation-only approaches made use of converters for transforming the outputs of Treebank parsers to inputs for realization, our approach instead employs a shared bidirectional grammar, so that the input to realization is guaranteed to be the same logical form constructed by the parser. In this regard, our approach is more similar to the ones pursued more recently by Carroll, Oepen and Velldal (2005; 2005; 2006), Nakanishi et al. (2005) and Cahill and van Genabith (2006) with HPSG and LFG grammars.

While we consider our approach to be the first to employ a supertagger for realization, or hypertagger, the approach is clearly reminiscent of the LTAG tree models of Srinivas and Rambow (2000). The main difference between the approaches is that ours consists of a multitagging step followed by the bottom-up construction of a realization chart, while theirs involves the top-down selection of the single most likely supertag for each node that is grammatically

compatible with the parent node, with the probability conditioned only on the child nodes. Note that although their approach does involve a subsequent lattice construction step, it requires making non-standard assumptions about the TAG; in contrast, ours follows the chart realization tradition of working with the same operations of grammatical combination as in parsing, including a well-defined notion of semantic composition. Additionally, as our tagger employs maximum entropy modeling, it is able to take into account a greater variety of contextual features, including those derived from parent nodes.

In comparison to other recent chart realization approaches, Nakanishi et al.'s is similar to ours in that it employs an iterative beam search, dynamically changing the beam size in order to cope with the large search space. However, their log-linear selection models have been adapted from ones used in parsing, and do not condition choices based on features of the input semantics to the same extent. In particular, while they employ a baseline maximum likelihood model that conditions the probability of a lexical entry upon its predicate argument structure (PAS) — that is, the set of elementary predications introduced by the lexical item — this probability does not take into account other elements of the local context, including parents and modifiers, and their lexical predicates. Similarly, Cahill and van Genabith condition the probability of their lexical rules on the set of feature-value pairs linked to the RHS of the rule, but do not take into account any additional context. Since their probabilistic models involve independence assumptions like those in a PCFG, and since they do not employ n-grams for scoring alternative realizations, their approach only keeps the single most likely edge in an equivalence class, rather than packing them into a forest. Carroll, Oepen and Velldal's approach is like Nakanishi et al.'s in that they adapt log-linear parsing models to the realization task; however, they employ manually written grammars on much smaller corpora, and perhaps for this reason they have not faced the need to employ an iterative beam search.

## 6 Conclusion

We have introduced a novel type of supertagger, which we have dubbed a *hypertagger*, that assigns CCG category labels to elementary predications in a structured semantic representation with high accuracy at several levels of tagging ambiguity in a fashion reminiscent of (Bangalore and Rambow, 2000). To our knowledge, we are the first to report tagging results in the semantic-to-syntactic direction. We have also shown that, by integrating this hypertagger with a broad-coverage CCG chart realizer, considerably faster realization times are possible (approximately twice as fast as compared with a realizer that performs simple lexical look-ups) with higher BLEU, METEOR and exact string match scores. Moreover, the hypertagger-augmented realizer finds more than twice the number of complete realizations, and further analysis revealed that the realization quality (as *per* modified BLEU and METEOR) is higher in the cases when the realizer finds a complete realization. This suggests that further improvements to the hypertagger will lead to more complete realizations, hence more high-quality realizations. Finally, further efforts to engineer a grammar suitable for realization from the CCGbank should provide richer feature sets, which, as our feature ablation study suggests, are useful for boosting hypertagging performance, hence for finding better and more complete realizations.

## Acknowledgements

The authors thank the anonymous reviewers, Chris Brew, Detmar Meurers and Eric Fosler-Lussier for helpful comments and discussion.

## References

- Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling CCG and Hybrid Logic Dependency Semantics. In *Proc. ACL-02*.
- Jason Baldridge and Geert-Jan Kruijff. 2003. Multi-Modal Combinatory Categorical Grammar. In *Proc. ACL-03*.
- Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorical Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.
- Srinivas Bangalore and Aravind K. Joshi. 1999. Su-

- per tagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2):237–265.
- Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proc. COLING-00*.
- Jeff Bilmes and Katrin Kirchhoff. 2003. Factored language models and general parallelized backoff. In *Proc. HLT-03*.
- Patrick Blackburn. 2000. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–625.
- Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *Proc. IWCS-6*.
- Stephen Boxwell and Michael White. 2008. Projecting Propbank roles onto the CCGbank. In *Proc. LREC-08*. To appear.
- Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proc. COLING-ACL '06*.
- Charles Callaway. 2003. Evaluating coverage for large symbolic NLG grammars. In *Proc. IJCAI-03*.
- John Carroll and Stefan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proc. IJCNLP-05*.
- John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznański. 1999. An efficient chart generator for (semi-) lexicalist grammars. In *Proc. ENLG-99*.
- Stephen Clark and James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).
- Stephen Clark. 2002. Supertagging for combinatory categorial grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)*, pages 19–24, Venice, Italy.
- James R. Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL-06)*, pages 697–704, Sydney, Australia.
- Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.
- Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh, Edinburgh, Scotland.
- Martin Kay. 1996. Chart generation. In *Proc. ACL-96*.
- Irene Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proc. INLG-02*.
- Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proc. NAACL-00*.
- Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of Workshop on Statistical Machine Translation at the 45th Annual Meeting of the Association of Computational Linguistics (ACL-2007)*, Prague.
- D C Liu and Jorge Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3).
- Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic methods for disambiguation of an HPSG-based chart generator. In *Proc. IWPT-05*.
- Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1).
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, PA.
- Carl J Pollard and Ivan A Sag. 1994. *Head-Driven Phrase Structure Grammar*. University Of Chicago Press.
- Yves Schabes, Anne Abeillé, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceedings of the 12<sup>th</sup> International Conference on Computational Linguistics (COLING-88)*, Budapest.
- Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, Massachusetts, USA.
- Andreas Stolcke. 2002. SRILM — An extensible language modeling toolkit. In *Proc. ICSLP-02*.
- Erik Velldal and Stephan Oepen. 2005. Maximum entropy models for realization ranking. In *Proc. MT Summit X*.
- Erik Velldal and Stephan Oepen. 2006. Statistical ranking in tactical generation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia, July.
- Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *Proc. of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT)*.
- Michael White. 2006a. CCG chart realization from disjunctive inputs. In *Proceedings, INLG 2006*.
- Michael White. 2006b. Efficient realization of coordinate structures in Combinatory Categorial Grammar. *Research on Language and Computation*, 4(1):39–75.

# Forest-Based Translation

Haitao Mi<sup>†</sup> Liang Huang<sup>‡</sup> Qun Liu<sup>†</sup>

<sup>†</sup>Key Lab. of Intelligent Information Processing  
Institute of Computing Technology  
Chinese Academy of Sciences  
P.O. Box 2704, Beijing 100190, China  
{htmi, liuqun}@ict.ac.cn

<sup>‡</sup>Department of Computer & Information Science  
University of Pennsylvania  
Levine Hall, 3330 Walnut Street  
Philadelphia, PA 19104, USA  
lhuang3@cis.upenn.edu

## Abstract

Among syntax-based translation models, the *tree-based* approach, which takes as input a parse tree of the source sentence, is a promising direction being faster and simpler than its string-based counterpart. However, current tree-based systems suffer from a major drawback: they only use the 1-best parse to direct the translation, which potentially introduces translation mistakes due to parsing errors. We propose a *forest-based* approach that translates a packed forest of exponentially many parses, which encodes many more alternatives than standard  $n$ -best lists. Large-scale experiments show an absolute improvement of 1.7 BLEU points over the 1-best baseline. This result is also 0.8 points higher than decoding with 30-best parses, and takes even less time.

## 1 Introduction

Syntax-based machine translation has witnessed promising improvements in recent years. Depending on the type of input, these efforts can be divided into two broad categories: the *string-based* systems whose input is a string to be simultaneously parsed and translated by a synchronous grammar (Wu, 1997; Chiang, 2005; Galley et al., 2006), and the *tree-based* systems whose input is already a parse tree to be directly converted into a target tree or string (Lin, 2004; Ding and Palmer, 2005; Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006). Compared with their string-based counterparts, tree-based systems offer some attractive features: they are much faster in decoding (linear time vs. cubic

time, see (Huang et al., 2006)), do not require a binary-branching grammar as in string-based models (Zhang et al., 2006), and can have separate grammars for parsing and translation, say, a context-free grammar for the former and a tree substitution grammar for the latter (Huang et al., 2006). However, despite these advantages, current tree-based systems suffer from a major drawback: they only use the 1-best parse tree to direct the translation, which potentially introduces translation mistakes due to parsing errors (Quirk and Corston-Oliver, 2006). This situation becomes worse with resource-poor source languages without enough Treebank data to train a high-accuracy parser.

One obvious solution to this problem is to take as input  $k$ -best parses, instead of a single tree. This  $k$ -best list postpones some disambiguation to the decoder, which may recover from parsing errors by getting a better translation from a non 1-best parse. However, a  $k$ -best list, with its limited scope, often has too few variations and too many redundancies; for example, a 50-best list typically encodes a combination of 5 or 6 binary ambiguities (since  $2^5 < 50 < 2^6$ ), and many subtrees are repeated across different parses (Huang, 2008). It is thus inefficient either to decode separately with each of these very similar trees. Longer sentences will also aggravate this situation as the number of parses grows exponentially with the sentence length.

We instead propose a new approach, *forest-based translation* (Section 3), where the decoder translates a *packed forest* of exponentially many parses,<sup>1</sup>

<sup>1</sup>There has been some confusion in the MT literature regarding the term *forest*: the word “forest” in “forest-to-string rules”

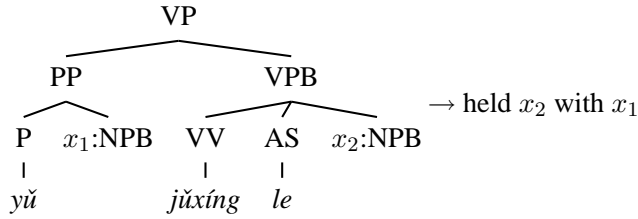


Figure 1: An example translation rule ( $r_3$  in Fig. 2).

which compactly encodes many more alternatives than  $k$ -best parses. This scheme can be seen as a compromise between the string-based and tree-based methods, while combining the advantages of both: decoding is still fast, yet does not commit to a single parse. Large-scale experiments (Section 4) show an improvement of 1.7 BLEU points over the 1-best baseline, which is also 0.8 points higher than decoding with 30-best trees, and takes even less time thanks to the sharing of common subtrees.

## 2 Tree-based systems

Current *tree-based* systems perform translation in two separate steps: parsing and decoding. A parser first parses the source language input into a 1-best tree  $T$ , and the decoder then searches for the best *derivation* (a sequence of translation steps)  $d^*$  that converts source tree  $T$  into a target-language string among all possible derivations  $D$ :

$$d^* = \arg \max_{d \in D} P(d|T). \quad (1)$$

We will now proceed with a running example translating from Chinese to English:

- (2) 布什 与 沙龙 举行了 会谈  
*Bùshí yǔ Shānlóng jǔxíng le huìtán*  
 Bush with/and Sharon<sub>1</sub> hold *pass.* talk<sub>2</sub>  
 “Bush held a talk<sub>2</sub> with Sharon<sub>1</sub>”

Figure 2 shows how this process works. The Chinese sentence (a) is first parsed into tree (b), which will be converted into an English string in 5 steps. First, at the root node, we apply rule  $r_1$  preserving top-level word-order between English and Chinese,

$$(r_1) \text{ IP}(x_1:\text{NPB } x_2:\text{VP}) \rightarrow x_1 x_2$$

(Liu et al., 2007) was a misnomer which actually refers to a set of several unrelated subtrees over disjoint spans, and should not be confused with the standard concept of *packed forest*.

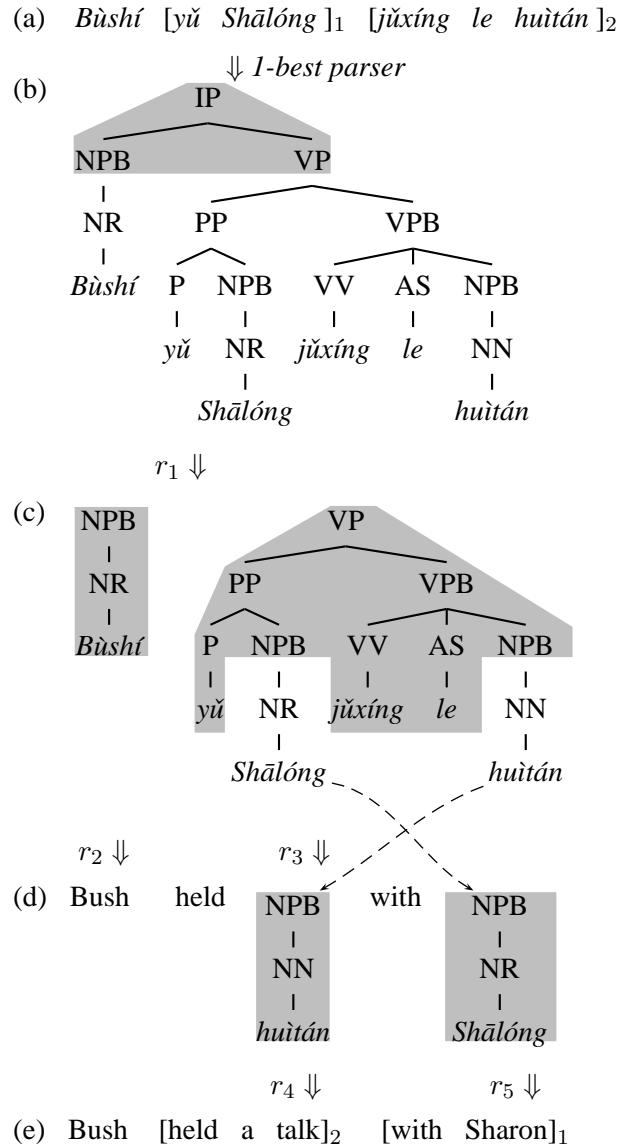


Figure 2: An example derivation of tree-to-string translation. Shaded regions denote parts of the tree that is pattern-matched with the rule being applied.

which results in two unfinished subtrees in (c). Then rule  $r_2$  grabs the *Bùshí* subtree and transliterate it

$$(r_2) \text{ NPB}(\text{NR}(\text{Bùshí})) \rightarrow \text{Bush.}$$

Similarly, rule  $r_3$  shown in Figure 1 is applied to the VP subtree, which swaps the two NPBs, yielding the situation in (d). This rule is particularly interesting since it has multiple levels on the source side, which has more expressive power than synchronous context-free grammars where rules are flat.

More formally, a (tree-to-string) **translation rule** (Huang et al., 2006) is a tuple  $\langle t, s, \phi \rangle$ , where  $t$  is the source-side tree, whose internal nodes are labeled by nonterminal symbols in  $N$ , and whose frontier nodes are labeled by source-side terminals in  $\Sigma$  or variables from a set  $\mathcal{X} = \{x_1, x_2, \dots\}$ ;  $s \in (\mathcal{X} \cup \Delta)^*$  is the target-side string where  $\Delta$  is the target language terminal set; and  $\phi$  is a mapping from  $\mathcal{X}$  to nonterminals in  $N$ . Each variable  $x_i \in \mathcal{X}$  occurs *exactly once* in  $t$  and *exactly once* in  $s$ . We denote  $\mathcal{R}$  to be the translation rule set. A similar formalism appears in another form in (Liu et al., 2006). These rules are in the reverse direction of the original string-to-tree transducer rules defined by Galley et al. (2004).

Finally, from step (d) we apply rules  $r_4$  and  $r_5$

( $r_4$ )  $\text{NPB}(\text{NN}(\text{huitán})) \rightarrow \text{a talk}$

( $r_5$ )  $\text{NPB}(\text{NR}(\text{Shālong})) \rightarrow \text{Sharon}$

which perform phrasal translations for the two remaining subtrees, respectively, and get the Chinese translation in (e).

### 3 Forest-based translation

We now extend the tree-based idea from the previous section to the case of forest-based translation. Again, there are two steps, parsing and decoding. In the former, a (modified) parser will parse the input sentence and output a packed forest (Section 3.1) rather than just the 1-best tree. Such a forest is usually huge in size, so we use the *forest pruning algorithm* (Section 3.4) to reduce it to a reasonable size. The pruned parse forest will then be used to direct the translation.

In the decoding step, we first convert the parse forest into a *translation forest* using the translation rule set, by similar techniques of pattern-matching from tree-based decoding (Section 3.2). Then the decoder searches for the best derivation on the translation forest and outputs the target string (Section 3.3).

#### 3.1 Parse Forest

Informally, a packed parse forest, or *forest* in short, is a compact representation of all the derivations (i.e., parse trees) for a given sentence under a context-free grammar (Billot and Lang, 1989). For

example, consider the Chinese sentence in Example (2) above, which has (at least) two readings depending on the part-of-speech of the word  $yǐ$ , which can be either a preposition (P “with”) or a conjunction (CC “and”). The parse tree for the preposition case is shown in Figure 2(b) as the 1-best parse, while for the conjunction case, the two proper nouns (*Bùshí* and *Shālong*) are combined to form a coordinated NP

$$\frac{\text{NPB}_{0,1} \quad \text{CC}_{1,2} \quad \text{NPB}_{2,3}}{\text{NP}_{0,3}} \quad (*)$$

which functions as the subject of the sentence. In this case the Chinese sentence is translated into

(3) “[Bush and Sharon] held a talk”.

Shown in Figure 3(a), these two parse trees can be represented as a single forest by sharing common subtrees such as  $\text{NPB}_{0,1}$  and  $\text{VPB}_{3,6}$ . Such a forest has a structure of a *hypergraph* (Klein and Manning, 2001; Huang and Chiang, 2005), where items like  $\text{NP}_{0,3}$  are called *nodes*, and deductive steps like (\*) correspond to *hyperedges*.

More formally, a **forest** is a pair  $\langle V, E \rangle$ , where  $V$  is the set of **nodes**, and  $E$  the set of **hyperedges**. For a given sentence  $w_{1:l} = w_1 \dots w_l$ , each node  $v \in V$  is in the form of  $X_{i,j}$ , which denotes the recognition of nonterminal  $X$  spanning the substring from positions  $i$  through  $j$  (that is,  $w_{i+1} \dots w_j$ ). Each hyperedge  $e \in E$  is a pair  $\langle \text{tails}(e), \text{head}(e) \rangle$ , where  $\text{head}(e) \in V$  is the *consequent node* in the deductive step, and  $\text{tails}(e) \in V^*$  is the list of *antecedent nodes*. For example, the hyperedge for deduction (\*) is notated:

$$\langle (\text{NPB}_{0,1}, \text{CC}_{1,2}, \text{NPB}_{2,3}), \text{NP}_{0,3} \rangle.$$

There is also a distinguished **root node** TOP in each forest, denoting the goal item in parsing, which is simply  $S_{0,l}$  where  $S$  is the start symbol and  $l$  is the sentence length.

#### 3.2 Translation Forest

Given a parse forest and a translation rule set  $\mathcal{R}$ , we can generate a *translation forest* which has a similar hypergraph structure. Basically, just as the depth-first traversal procedure in tree-based decoding (Figure 2), we visit in top-down order each node  $v$  in the

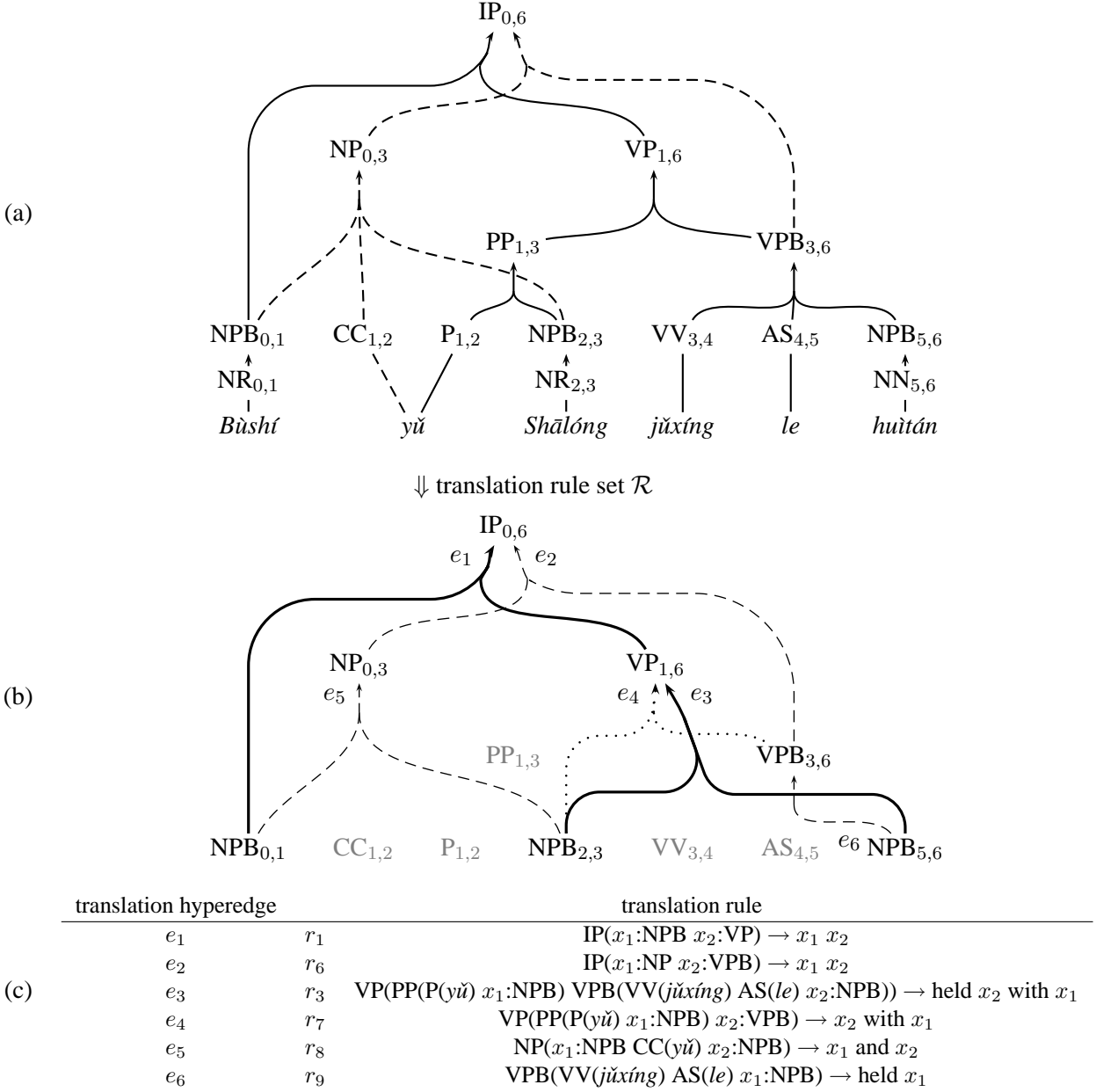


Figure 3: (a) the parse forest of the example sentence; solid hyperedges denote the 1-best parse in Figure 2(b) while dashed hyperedges denote the alternative parse due to Deduction (\*). (b) the corresponding translation forest after applying the translation rules (lexical rules not shown); the derivation shown in bold solid lines ( $e_1$  and  $e_3$ ) corresponds to the derivation in Figure 2; the one shown in dashed lines ( $e_2$ ,  $e_5$ , and  $e_6$ ) uses the alternative parse and corresponds to the translation in Example (3). (c) the correspondence between translation hyperedges and translation rules.

parse forest, and try to pattern-match each translation rule  $r$  against the local sub-forest under node  $v$ . For example, in Figure 3(a), at node  $VP_{1,6}$ , two rules  $r_3$  and  $r_7$  both matches the local subforest, and will thus generate two *translation hyperedges*  $e_3$  and  $e_4$  (see Figure 3(b-c)).

More formally, we define a function  $match(r, v)$  which attempts to pattern-match rule  $r$  at node  $v$  in the parse forest, and in case of success, returns a list of descendent nodes of  $v$  that are matched to the variables in  $r$ , or returns an empty list if the match fails. Note that this procedure is recursive and may

---

**Pseudocode 1** The conversion algorithm.

---

```
1: Input: parse forest  $H_p$  and rule set  $\mathcal{R}$ 
2: Output: translation forest  $H_t$ 
3: for each node  $v \in V_p$  in top-down order do
4:   for each translation rule  $r \in \mathcal{R}$  do
5:      $vars \leftarrow match(r, v)$   $\triangleright$  variables
6:     if  $vars$  is not empty then
7:        $e \leftarrow \langle vars, v, s(r) \rangle$ 
8:       add translation hyperedge  $e$  to  $H_t$ 
```

---

involve multiple parse hyperedges. For example,

$$match(r_3, VP_{1,6}) = (NPB_{2,3}, NPB_{5,6}),$$

which covers three parse hyperedges, while nodes in gray do not pattern-match any rule (although they are involved in the matching of other nodes, where they match *interior nodes* of the source-side tree fragments in a rule). We can thus construct a translation hyperedge from  $match(r, v)$  to  $v$  for each node  $v$  and rule  $r$ . In addition, we also need to keep track of the *target string*  $s(r)$  specified by rule  $r$ , which includes target-language terminals and variables. For example,  $s(r_3) = \text{“held } x_2 \text{ with } x_1\text{”}$ . The subtranslations of the matched variable nodes will be substituted for the variables in  $s(r)$  to get a complete translation for node  $v$ . So a translation hyperedge  $e$  is a triple  $\langle tails(e), head(e), s \rangle$  where  $s$  is the target string from the rule, for example,

$$e_3 = \langle (NPB_{2,3}, NPB_{5,6}), VP_{1,6}, \text{“held } x_2 \text{ with } x_1\text{”} \rangle.$$

This procedure is summarized in Pseudocode 1.

### 3.3 Decoding Algorithms

The decoder performs two tasks on the translation forest: 1-best search with integrated language model (LM), and  $k$ -best search with LM to be used in minimum error rate training. Both tasks can be done efficiently by forest-based algorithms based on  $k$ -best parsing (Huang and Chiang, 2005).

For 1-best search, we use the *cube pruning* technique (Chiang, 2007; Huang and Chiang, 2007) which approximately intersects the translation forest with the LM. Basically, cube pruning works bottom up in a forest, keeping at most  $k$  +LM items at each node, and uses the best-first expansion idea from the Algorithm 2 of Huang and Chiang (2005) to speed

up the computation. An +LM item of node  $v$  has the form  $(v^{a*b})$ , where  $a$  and  $b$  are the target-language *boundary words*. For example,  $(VP_{1,6}^{\text{held } * \text{Sharon}})$  is an +LM item with its translation starting with “held” and ending with “Sharon”. This scheme can be easily extended to work with a general  $n$ -gram by storing  $n - 1$  words at both ends (Chiang, 2007).

For  $k$ -best search after getting 1-best derivation, we use the lazy Algorithm 3 of Huang and Chiang (2005) that works backwards from the root node, incrementally computing the second, third, through the  $k$ th best alternatives. However, this time we work on a finer-grained forest, called *translation+LM* forest, resulting from the intersection of the translation forest and the LM, with its nodes being the +LM items during cube pruning. Although this new forest is prohibitively large, Algorithm 3 is very efficient with minimal overhead on top of 1-best.

### 3.4 Forest Pruning Algorithm

We use the pruning algorithm of (Jonathan Graehl, p.c.; Huang, 2008) that is very similar to the method based on marginal probability (Charniak and Johnson, 2005), except that it prunes hyperedges as well as nodes. Basically, we use an Inside-Outside algorithm to compute the Viterbi inside cost  $\beta(v)$  and the Viterbi outside cost  $\alpha(v)$  for each node  $v$ , and then compute the **merit**  $\alpha\beta(e)$  for each hyperedge:

$$\alpha\beta(e) = \alpha(head(e)) + \sum_{u_i \in tails(e)} \beta(u_i) \quad (4)$$

Intuitively, this merit is the cost of the best derivation that traverses  $e$ , and the difference  $\delta(e) = \alpha\beta(e) - \beta(\text{TOP})$  can be seen as the distance away from the globally best derivation. We prune away a hyperedge  $e$  if  $\delta(e) > p$  for a threshold  $p$ . Nodes with all incoming hyperedges pruned are also pruned.

## 4 Experiments

We can extend the simple model in Equation 1 to a log-linear one (Liu et al., 2006; Huang et al., 2006):

$$d^* = \arg \max_{d \in D} P(d | T)^{\lambda_0} \cdot e^{\lambda_1 |d|} \cdot P_{\text{lm}}(s)^{\lambda_2} \cdot e^{\lambda_3 |s|} \quad (5)$$

where  $T$  is the 1-best parse,  $e^{\lambda_1 |d|}$  is the penalty term on the number of rules in a derivation,  $P_{\text{lm}}(s)$  is the language model and  $e^{\lambda_3 |s|}$  is the length penalty term



on target translation. The derivation probability conditioned on 1-best tree,  $P(d | T)$ , should now be replaced by  $P(d | H_p)$  where  $H_p$  is the parse forest, which decomposes into the product of probabilities of translation rules  $r \in d$ :

$$P(d | H_p) = \prod_{r \in d} P(r) \quad (6)$$

where each  $P(r)$  is the product of five probabilities:

$$P(r) = P(t | s)^{\lambda_4} \cdot P_{\text{lex}}(t | s)^{\lambda_5} \cdot P(s | t)^{\lambda_6} \cdot P_{\text{lex}}(s | t)^{\lambda_7} \cdot P(t | H_p)^{\lambda_8}. \quad (7)$$

Here  $t$  and  $s$  are the source-side tree and target-side string of rule  $r$ , respectively,  $P(t | s)$  and  $P(s | t)$  are the two translation probabilities, and  $P_{\text{lex}}(\cdot)$  are the lexical probabilities. The only extra term in forest-based decoding is  $P(t | H_p)$  denoting the source side parsing probability of the current translation rule  $r$  in the parse forest, which is the product of probabilities of each parse hyperedge  $e_p$  covered in the pattern-match of  $t$  against  $H_p$  (which can be recorded at conversion time):

$$P(t | H_p) = \prod_{e_p \in H_p, e_p \text{ covered by } t} P(e_p). \quad (8)$$

#### 4.1 Data preparation

Our experiments are on Chinese-to-English translation, and we use the Chinese parser of Xiong et al. (2005) to parse the source side of the bitext. Following Huang (2008), we modify the parser to output a packed forest for each sentence.

Our training corpus consists of 31,011 sentence pairs with 0.8M Chinese words and 0.9M English words. We first word-align them by GIZA++ refined by “diagand” from Koehn et al. (2003), and apply the tree-to-string rule extraction algorithm (Galley et al., 2006; Liu et al., 2006), which resulted in 346K translation rules. Note that our rule extraction is still done on 1-best parses, while decoding is on  $k$ -best parses or packed forests. We also use the SRI Language Modeling Toolkit (Stolcke, 2002) to train a trigram language model with Kneser-Ney smoothing on the English side of the bitext.

We use the 2002 NIST MT Evaluation test set as our development set (878 sentences) and the 2005

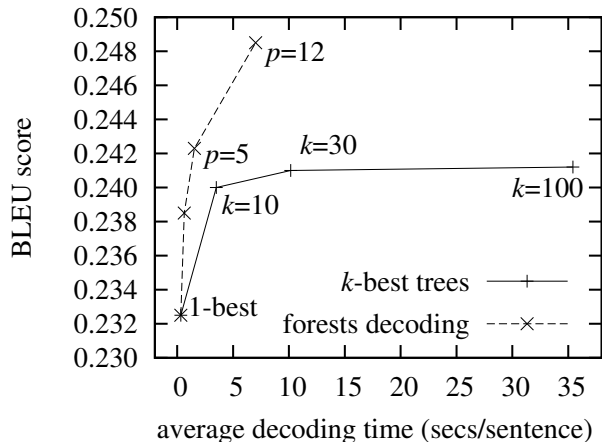


Figure 4: Comparison of decoding on forests with decoding on  $k$ -best trees.

NIST MT Evaluation test set as our test set (1082 sentences), with on average 28.28 and 26.31 words per sentence, respectively. We evaluate the translation quality using the *case-sensitive* BLEU-4 metric (Papineni et al., 2002). We use the standard minimum error-rate training (Och, 2003) to tune the feature weights to maximize the system’s BLEU score on the dev set. On dev and test sets, we prune the Chinese parse forests by the forest pruning algorithm in Section 3.4 with a threshold of  $p = 12$ , and then convert them into translation forests using the algorithm in Section 3.2. To increase the coverage of the rule set, we also introduce a *default translation hyperedge* for each parse hyperedge by monotonically translating each tail node, so that we can always at least get a complete translation in the end.

#### 4.2 Results

The BLEU score of the baseline 1-best decoding is 0.2325, which is consistent with the result of 0.2302 in (Liu et al., 2007) on the same training, development and test sets, and with the same rule extraction procedure. The corresponding BLEU score of Pharaoh (Koehn, 2004) is 0.2182 on this dataset.

Figure 4 compares forest decoding with decoding on  $k$ -best trees in terms of speed and quality. Using more than one parse tree apparently improves the BLEU score, but at the cost of much slower decoding, since each of the top- $k$  trees has to be decoded individually although they share many common subtrees. Forest decoding, by contrast, is much faster

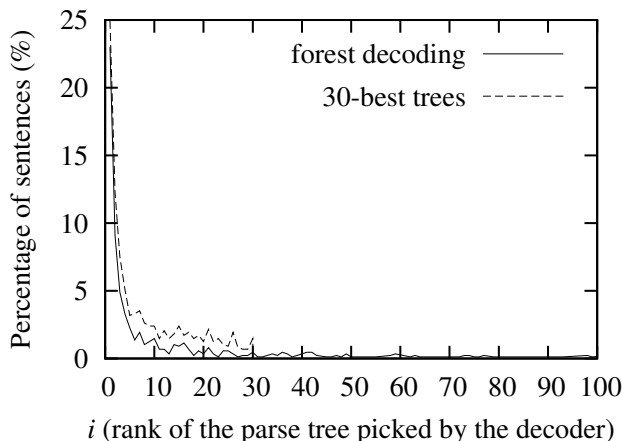


Figure 5: Percentage of the  $i$ -th best parse tree being picked in decoding. 32% of the distribution for forest decoding is beyond top-100 and is not shown on this plot.

and produces consistently better BLEU scores. With pruning threshold  $p = 12$ , it achieved a BLEU score of 0.2485, which is an absolute improvement of 1.6% points over the 1-best baseline, and is statistically significant using the *sign-test* of Collins et al. (2005) ( $p < 0.01$ ).

We also investigate the question of how often the  $i$ th-best parse tree is picked to direct the translation ( $i = 1, 2, \dots$ ), in both  $k$ -best and forest decoding schemes. A packed forest can be roughly viewed as a (virtual)  $\infty$ -best list, and we can thus ask how often is a parse beyond top- $k$  used by a forest, which relates to the fundamental limitation of  $k$ -best lists. Figure 5 shows that, the 1-best parse is still preferred 25% of the time among 30-best trees, and 23% of the time by the forest decoder. These ratios decrease dramatically as  $i$  increases, but the forest curve has a much longer tail in large  $i$ . Indeed, 40% of the trees preferred by a forest is beyond top-30, 32% is beyond top-100, and even 20% beyond top-1000. This confirms the fact that we need exponentially large  $k$ -best lists with the explosion of alternatives, whereas a forest can encode these information compactly.

### 4.3 Scaling to large data

We also conduct experiments on a larger dataset, which contains 2.2M training sentence pairs. Besides the trigram language model trained on the English side of these bitext, we also use another trigram model trained on the first 1/3 of the Xinhua portion of Gigaword corpus. The two LMs have dis-

approach \ ruleset	TR	TR+BP
1-best tree	0.2666	0.2939
30-best trees	0.2755	0.3084
forest ( $p = 12$ )	<b>0.2839</b>	<b>0.3149</b>

Table 1: BLEU score results from training on large data.

tinct weights tuned by minimum error rate training. The dev and test sets remain the same as above.

Furthermore, we also make use of bilingual phrases to improve the coverage of the ruleset. Following Liu et al. (2006), we prepare a phrase-table from a phrase-extractor, e.g. Pharaoh, and at decoding time, for each node, we construct on-the-fly flat translation rules from phrases that match the source-side span of the node. These phrases are called *syntactic phrases* which are consistent with syntactic constituents (Chiang, 2005), and have been shown to be helpful in tree-based systems (Galley et al., 2006; Liu et al., 2006).

The final results are shown in Table 1, where TR denotes translation rule only, and TR+BP denotes the inclusion of bilingual phrases. The BLEU score of forest decoder with TR is 0.2839, which is a 1.7% points improvement over the 1-best baseline, and this difference is statistically significant ( $p < 0.01$ ). Using bilingual phrases further improves the BLEU score by 3.1% points, which is 2.1% points higher than the respective 1-best baseline. We suspect this larger improvement is due to the alternative constituents in the forest, which activates many syntactic phrases suppressed by the 1-best parse.

## 5 Conclusion and future work

We have presented a novel forest-based translation approach which uses a packed forest rather than the 1-best parse tree (or  $k$ -best parse trees) to direct the translation. Forest provides a compact data-structure for efficient handling of exponentially many tree structures, and is shown to be a promising direction with state-of-the-art translation results and reasonable decoding speed. This work can thus be viewed as a compromise between string-based and tree-based paradigms, with a good trade-off between speed and accuracy. For future work, we would like to use packed forests not only in decoding, but also for translation rule extraction during training.

## Acknowledgement

Part of this work was done while L. H. was visiting CAS/ICT. The authors were supported by National Natural Science Foundation of China, Contracts 60736014 and 60573188, and 863 State Key Project No. 2006AA010108 (H. M and Q. L.), and by NSF ITR EIA-0205456 (L. H.). We would also like to thank Chris Quirk for inspirations, Yang Liu for help with rule extraction, Mark Johnson for posing the question of virtual  $\infty$ -best list, and the anonymous reviewers for suggestions.

## References

- Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of ACL '89*, pages 143–151.
- Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine-grained  $n$ -best parsing and discriminative reranking. In *Proceedings of the 43rd ACL*.
- David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, Michigan, June.
- David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228.
- Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Ann Arbor, Michigan, June.
- Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL*, pages 541–548, Ann Arbor, Michigan, June.
- Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What’s in a translation rule? In *HLT-NAACL*, pages 273–280, Boston, MA.
- Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL*, pages 961–968, Sydney, Australia, July.
- Liang Huang and David Chiang. 2005. Better  $k$ -best parsing. In *Proceedings of Ninth International Workshop on Parsing Technologies (IWPT-2005)*, Vancouver, Canada.
- Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151, Prague, Czech Republic, June.
- Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*, Boston, MA, August.
- Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL*, Columbus, OH.
- Dan Klein and Christopher D. Manning. 2001. Parsing and Hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001)*, 17-19 October 2001, Beijing, China.
- Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, Edmonton, AB, Canada.
- Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*, pages 115–124.
- Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of the 20th COLING*, Barcelona, Spain.
- Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING-ACL*, pages 609–616, Sydney, Australia, July.
- Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proceedings of ACL*, pages 704–711, Prague, Czech Republic, June.
- Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, USA, July.
- Chris Quirk and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *Proceedings of EMNLP*.
- Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of ACL*, pages 271–279, Ann Arbor, Michigan, June.
- Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.
- Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.
- Deyi Xiong, Shuanglong Li, Qun Liu, and Shouxun Lin. 2005. Parsing the Penn Chinese Treebank with semantic knowledge. In *Proceedings of IJCNLP 2005*, pages 70–81, Jeju Island, South Korea.
- Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of HLT-NAACL*, New York, NY.

# A Discriminative Latent Variable Model for Statistical Machine Translation

Phil Blunsom, Trevor Cohn and Miles Osborne  
School of Informatics, University of Edinburgh  
2 Buccleuch Place, Edinburgh, EH8 9LW, UK  
{pblunsom, tcohn, miles}@inf.ed.ac.uk

## Abstract

Large-scale discriminative machine translation promises to further the state-of-the-art, but has failed to deliver convincing gains over current heuristic frequency count systems. We argue that a principle reason for this failure is not dealing with multiple, equivalent translations. We present a translation model which models derivations as a latent variable, in both training and decoding, and is fully discriminative and globally optimised. Results show that accounting for multiple derivations does indeed improve performance. Additionally, we show that regularisation is essential for maximum conditional likelihood models in order to avoid degenerate solutions.

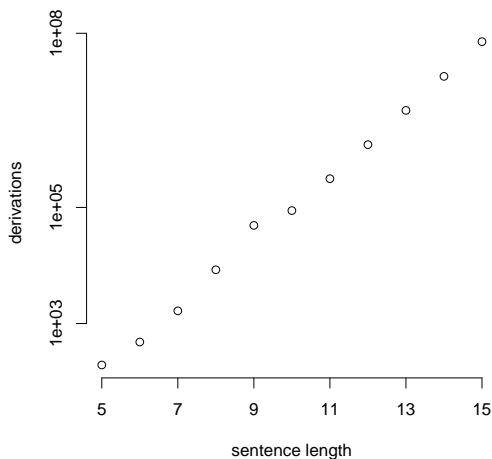
## 1 Introduction

Statistical machine translation (SMT) has seen a resurgence in popularity in recent years, with progress being driven by a move to phrase-based and syntax-inspired approaches. Progress within these approaches however has been less dramatic. We believe this is because these frequency count based<sup>1</sup> models cannot easily incorporate non-independent and overlapping features, which are extremely useful in describing the translation process. Discriminative models of translation can include such features without making assumptions of independence or explicitly modelling their interdependence. However, while discriminative models promise much, they have not been shown to deliver significant gains

<sup>1</sup>We class approaches using minimum error rate training (Och, 2003) *frequency count based* as these systems re-scale a handful of generative features estimated from frequency counts and do not support large sets of non-independent features.

over their simpler cousins. We argue that this is due to a number of inherent problems that discriminative models for SMT must address, in particular the problems of spurious ambiguity and degenerate solutions. These occur when there are many ways to translate a source sentence to the same target sentence by applying a sequence of steps (a *derivation*) of either phrase translations or synchronous grammar rules, depending on the type of system. Existing discriminative models require a reference derivation to optimise against, however no parallel corpora annotated for derivations exist. Ideally, a model would account for this ambiguity by marginalising out the derivations, thus predicting the best *translation* rather than the best *derivation*. However, doing so exactly is NP-complete. For this reason, to our knowledge, all discriminative models proposed to date either side-step the problem by choosing simple model and feature structures, such that spurious ambiguity is lessened or removed entirely (Ittycheriah and Roukos, 2007; Watanabe et al., 2007), or else ignore the problem and treat derivations as translations (Liang et al., 2006; Tillmann and Zhang, 2007).

In this paper we directly address the problem of spurious ambiguity in discriminative models. We use a synchronous context free grammar (SCFG) translation system (Chiang, 2007), a model which has yielded state-of-the-art results on many translation tasks. We present two main contributions. First, we develop a log-linear model of translation which is globally trained on a significant number of parallel sentences. This model maximises the conditional likelihood of the data,  $p(\mathbf{e}|\mathbf{f})$ , where  $\mathbf{e}$  and  $\mathbf{f}$  are the English and foreign sentences, respectively. Our estimation method is theoretically sound, avoiding the biases of the heuristic relative frequency estimates



**Figure 1.** Exponential relationship between sentence length and the average number of derivations (on a log scale) for each reference sentence in our training corpus.

(Koehn et al., 2003). Second, within this framework, we model the derivation,  $\mathbf{d}$ , as a latent variable,  $p(\mathbf{e}, \mathbf{d}|\mathbf{f})$ , which is marginalised out in training and decoding. We show empirically that this treatment results in significant improvements over a maximum-derivation model.

The paper is structured as follows. In Section 2 we list the challenges that discriminative SMT must face above and beyond the current systems. We situate our work, and previous work, on discriminative systems in this context. We present our model in Section 3, including our means of training and decoding. Section 4 reports our experimental setup and results, and finally we conclude in Section 5.

## 2 Challenges for Discriminative SMT

Discriminative models allow for the use of expressive features, in the order of thousands or millions, which can reference arbitrary aspects of the source sentence. Given most successful SMT models have a highly lexicalised grammar (or grammar equivalent), these features can be used to smuggle in linguistic information, such as syntax and document context. With this undoubted advantage come four major challenges when compared to standard frequency count SMT models:

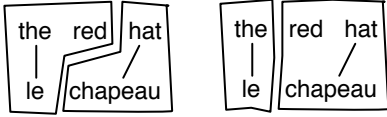
1. There is no one reference derivation. Often there are thousands of ways of translating a source sentence into the reference translation. Figure 1 illustrates the exponential relationship

between sentence length and the number of derivations. Training is difficult without a clear target, and predicting only one derivation at test time is fraught with danger.

2. Parallel translation data is often very noisy, with such problems as non-literal translations, poor sentence- and word-alignments. A model which exactly translates the training data will inevitably perform poorly on held-out data. This problem of over-fitting is exacerbated in discriminative models with large, expressive, feature sets. Regularisation is essential for models with more than a handful of features.
3. Learning with a large feature set requires many training examples and typically many iterations of a solver during training. While current models focus solely on efficient decoding, discriminative models must also allow for efficient training.

Past work on discriminative SMT only address some of these problems. To our knowledge no systems directly address Problem 1, instead choosing to ignore the problem by using one or a small handful of reference derivations in an n-best list (Liang et al., 2006; Watanabe et al., 2007), or else making local independence assumptions which side-step the issue (Ittycheriah and Roukos, 2007; Tillmann and Zhang, 2007; Wellington et al., 2006). These systems all include regularisation, thereby addressing Problem 2. An interesting counterpoint is the work of DeNero et al. (2006), who show that their unregularised model finds degenerate solutions. Some of these discriminative systems have been trained on large training sets (Problem 3); these systems are the local models, for which training is much simpler. Both the global models (Liang et al., 2006; Watanabe et al., 2007) use fairly small training sets, and there is no evidence that their techniques will scale to larger data sets.

Our model addresses all three of the above problems within a global model, without resorting to n-best lists or local independence assumptions. Furthermore, our model explicitly accounts for spurious ambiguity without altering the model structure or arbitrarily selecting one derivation. Instead we model the translation distribution with a latent variable for the derivation, which we marginalise out in training and decoding.



**Figure 2.** The dropping of an adjective in this example means that there is no one segmentation that we could choose that would allow a system to learn  $le \rightarrow the$  and  $chapeau \rightarrow hat$ .

$$\begin{aligned}
 \langle S \rangle &\rightarrow \langle S_{\square} X_{\square}, S_{\square} X_{\square} \rangle \\
 \langle S \rangle &\rightarrow \langle X_{\square}, X_{\square} \rangle \\
 \langle X \rangle &\rightarrow \langle ne X_{\square} pas, does not X_{\square} \rangle \\
 \langle X \rangle &\rightarrow \langle va, go \rangle \\
 \langle X \rangle &\rightarrow \langle il, he \rangle
 \end{aligned}$$

**Figure 3.** A simple SCFG, with non-terminal symbols  $S$  and  $X$ , which performs the transduction:  $il ne vas pas \Rightarrow he does not go$

This itself provides robustness to noisy data, in addition to the explicit regularisation from a prior over the model parameters. For example, in many cases there is no one perfect derivation, but rather many imperfect ones which each include some good translation fragments. The model can learn from many of these derivations and thereby learn from all these translation fragments. This situation is illustrated in Figure 2 where the non-translated adjective *red* means neither segmentation is ‘correct’, although both together present positive evidence for the two lexical translations.

We present efficient methods for training and prediction, demonstrating their scaling properties by training on more than a hundred thousand training sentences. Finally, we stress that our main findings are general ones. These results could – and should – be applied to other models, discriminative and generative, phrase- and syntax-based, to further progress the state-of-the-art in machine translation.

### 3 Discriminative Synchronous Transduction

A synchronous context free grammar (SCFG) consists of paired CFG rules with co-indexed non-terminals (Lewis II and Stearns, 1968). By assigning the source and target languages to the respective sides of a SCFG it is possible to describe translation as the process of parsing the source sentence using a CFG, while generating the target translation from

the other (Chiang, 2007). All the models we present use the grammar extraction technique described in Chiang (2007), and are bench-marked against our own implementation of this hierarchical model (Hiero). Figure 3 shows a simple instance of a hierarchical grammar with two non-terminals. Note that our approach is general and could be used with other synchronous grammar transducers (e.g., Galley et al. (2006)).

#### 3.1 A global log-linear model

Our log-linear translation model defines a conditional probability distribution over the target translations of a given source sentence. A particular sequence of SCFG rule applications which produces a translation from a source sentence is referred to as a *derivation*, and each translation may be produced by many different derivations. As the training data only provides source and target sentences, the derivations are modelled as a latent variable.

The conditional probability of a derivation,  $\mathbf{d}$ , for a target translation,  $\mathbf{e}$ , conditioned on the source,  $\mathbf{f}$ , is given by:

$$p_{\Lambda}(\mathbf{d}, \mathbf{e} | \mathbf{f}) = \frac{\exp \sum_k \lambda_k H_k(\mathbf{d}, \mathbf{e}, \mathbf{f})}{Z_{\Lambda}(\mathbf{f})} \quad (1)$$

$$\text{where } H_k(\mathbf{d}, \mathbf{e}, \mathbf{f}) = \sum_{r \in \mathbf{d}} h_k(\mathbf{f}, r) \quad (2)$$

Here  $k$  ranges over the model’s features, and  $\Lambda = \{\lambda_k\}$  are the model parameters (weights for their corresponding features). The feature functions  $H_k$  are predefined real-valued functions over the source and target sentences, and can include overlapping and non-independent features of the data. The features must decompose with the derivation, as shown in (2). The features can reference the entire source sentence coupled with each rule,  $r$ , in a derivation. The distribution is globally normalised by the partition function,  $Z_{\Lambda}(\mathbf{f})$ , which sums out the numerator in (1) for every derivation (and therefore every translation) of  $\mathbf{f}$ :

$$Z_{\Lambda}(\mathbf{f}) = \sum_{\mathbf{e}} \sum_{\mathbf{d} \in \Delta(\mathbf{e}, \mathbf{f})} \exp \sum_k \lambda_k H_k(\mathbf{d}, \mathbf{e}, \mathbf{f})$$

Given (1), the conditional probability of a target translation given the source is the sum over all of its derivations:

$$p_{\Lambda}(\mathbf{e} | \mathbf{f}) = \sum_{\mathbf{d} \in \Delta(\mathbf{e}, \mathbf{f})} p_{\Lambda}(\mathbf{d}, \mathbf{e} | \mathbf{f}) \quad (3)$$

where  $\Delta(\mathbf{e}, \mathbf{f})$  is the set of all derivations of the target sentence  $\mathbf{e}$  from the source  $\mathbf{f}$ .

Most prior work in SMT, both generative and discriminative, has approximated the sum over derivations by choosing a single ‘best’ derivation using a Viterbi or beam search algorithm. In this work we show that it is both tractable and desirable to directly account for derivational ambiguity. Our findings echo those observed for latent variable log-linear models successfully used in monolingual parsing (Clark and Curran, 2007; Petrov et al., 2007). These models marginalise over derivations leading to a dependency structure and splits of non-terminal categories in a PCFG, respectively.

### 3.2 Training

The parameters of our model are estimated from our training sample using a maximum *a posteriori* (MAP) estimator. This maximises the likelihood of the parallel training sentences,  $\mathcal{D} = \{(\mathbf{e}, \mathbf{f})\}$ , penalised using a prior, i.e.,  $\Lambda^{MAP} = \arg \max_{\Lambda} p_{\Lambda}(\mathcal{D})p(\Lambda)$ . We use a zero-mean Gaussian prior with the probability density function  $p_0(\lambda_k) \propto \exp(-\lambda_k^2/2\sigma^2)$ .<sup>2</sup> This results in the following log-likelihood objective and corresponding gradient:

$$\mathcal{L} = \sum_{(\mathbf{e}, \mathbf{f}) \in \mathcal{D}} \log p_{\Lambda}(\mathbf{e}|\mathbf{f}) + \sum_k \log p_0(\lambda_k) \quad (4)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = E_{p_{\Lambda}(\mathbf{d}|\mathbf{e}, \mathbf{f})}[h_k] - E_{p_{\Lambda}(\mathbf{e}|\mathbf{f})}[h_k] - \frac{\lambda_k}{\sigma^2} \quad (5)$$

In order to train the model, we maximise equation (4) using L-BFGS (Malouf, 2002; Sha and Pereira, 2003). This method has been demonstrated to be effective for (non-convex) log-linear models with latent variables (Clark and Curran, 2004; Petrov et al., 2007). Each L-BFGS iteration requires the objective value and its gradient with respect to the model parameters. These are calculated using inside-outside inference over the feature forest defined by the SCFG parse chart of  $\mathbf{f}$  yielding the partition function,  $Z_{\Lambda}(\mathbf{f})$ , required for the log-likelihood, and the marginals, required for its derivatives.

Efficiently calculating the objective and its gradient requires two separate packed charts, each representing a derivation forest. The first one is the full chart over the space of possible derivations given the

<sup>2</sup>In general, any conjugate prior could be used instead of a simple Gaussian.

source sentence. The inside-outside algorithm over this chart gives the marginal probabilities for each chart cell, from which we can find the feature expectations. The second chart contains the space of derivations which produce the reference translation from the source. The derivations in this chart are a subset of those in the full derivation chart. Again, we use the inside-outside algorithm to find the ‘reference’ feature expectations from this chart. These expectations are analogous to the empirical observation of maximum entropy classifiers.

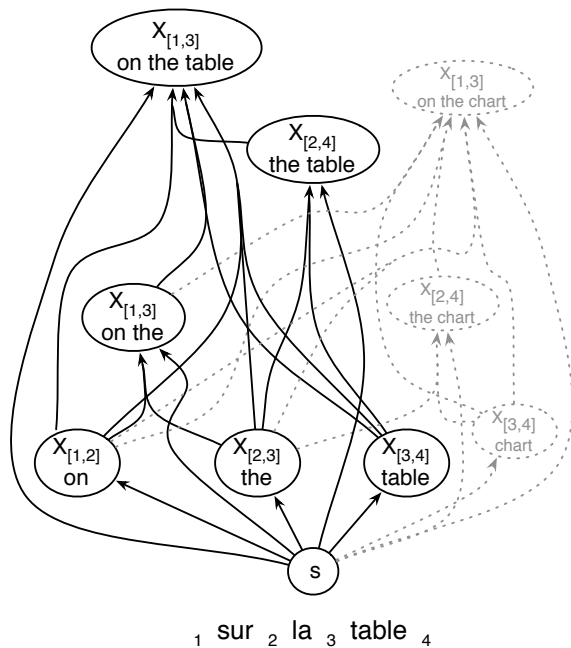
Given these two charts we can calculate the log-likelihood of the reference translation as the inside-score from the sentence spanning cell of the reference chart, normalised by the inside-score of the spanning cell from the full chart. The gradient is calculated as the difference of the feature expectations of the two charts. Clark and Curran (2004) provides a more complete discussion of parsing with a log-linear model and latent variables.

The full derivation chart is produced using a CYK parser in the same manner as Chiang (2005), and has complexity  $O(|\mathbf{e}|^3)$ . We produce the reference chart by synchronously parsing the source and reference sentences using a variant of CYK algorithm over two dimensions, with a time complexity of  $O(|\mathbf{e}|^3|\mathbf{f}|^3)$ . This is an instance of the ITG alignment algorithm (Wu, 1997). This step requires the reference translation for each training instance to be contained in the model’s hypothesis space. Achieving full coverage implies inducing a grammar which generates all observed source-target pairs, which is difficult in practice. Instead we discard the unreachable portion of the training sample (24% in our experiments). The proportion of discarded sentences is a function of the grammar used. Extraction heuristics other than the method used herein (Chiang, 2007) could allow complete coverage (e.g., Galley et al. (2004)).

### 3.3 Decoding

Accounting for all derivations of a given translation should benefit not only training, but also decoding. Unfortunately marginalising over derivations in decoding is NP-complete. The standard solution is to approximate the maximum probability translation using a single derivation (Koehn et al., 2003).

Here we approximate the sum over derivations directly using a beam search in which we produce a beam of high probability translation sub-strings for each cell in the parse chart. This algorithm is sim-



**Figure 4.** Hypergraph representation of max translation decoding. Each chart cell must store the entire target string generated.

ilar to the methods for decoding with a SCFG intersected with an  $n$ -gram language model, which require language model contexts to be stored in each chart cell. However, while Chiang (2005) stores an abbreviated context composed of the  $n - 1$  target words on the left and right edge of the target substring, here we store the entire target string. Additionally, instead of maximising scores in each beam cell, we sum the inside scores for each derivation that produces a given string for that cell. When the beam search is complete we have a list of translations in the top beam cell spanning the entire source sentence along with their approximated inside derivation scores. Thus we can assign each translation string a probability by normalising its inside score by the sum of the inside scores of all the translations spanning the entire sentence.

Figure 4 illustrates the search process for the simple grammar from Table 2. Each graph node represents a hypothesis translation substring covering a sub-span of the source string. The space of translation sub-strings is exponential in each cell’s span, and our algorithm can only sum over a small fraction of the possible strings. Therefore the resulting probabilities are only estimates. However, as demonstrated in Section 4, this algorithm is considerably more effective than maximum derivation (Viterbi) decoding.

## 4 Evaluation

Our model evaluation was motivated by the following questions: (1) the effect of maximising translations rather than derivations in training and decoding; (2) whether a regularised model performs better than a maximum likelihood model; (3) how the performance of our model compares with a frequency count based hierarchical system; and (4) how translation performance scales with the number of training examples.

We performed all of our experiments on the Europarl V2 French-English parallel corpus.<sup>3</sup> The training data was created by filtering the full corpus for all the French sentences between five and fifteen words in length, resulting in 170K sentence pairs. These limits were chosen as a compromise between experiment turnaround time and leaving a large enough corpus to obtain indicative results. The development and test data was taken from the 2006 NAACL and 2007 ACL workshops on machine translation, also filtered for sentence length.<sup>4</sup> Tuning of the regularisation parameter and MERT training of the benchmark models was performed on *dev2006*, while the test set was the concatenation of *devtest2006*, *test2006* and *test2007*, amounting to 315 development and 1164 test sentences.

Here we focus on evaluating our model’s basic ability to learn a conditional distribution from simple binary features, directly comparable to those currently employed in frequency count models. As such, our base model includes a single binary identity feature per-rule, equivalent to the  $p(e|f)$  parameters defined on each rule in standard models.

As previously noted, our model must be able to derive the reference sentence from the source for it to be included in training. For both our discriminative and benchmark (Hiero) we extracted our grammar on the 170K sentence corpus using the approach described in Chiang (2007), resulting in 7.8 million rules. The discriminative model was then trained on the training partition, however only 130K of the sentences were used as the model could not produce a derivation of the reference for the remaining sentences. There were many grammar rules that the discriminative model did not observe in a reference derivation, and thus could not assign their feature a positive weight. While the benchmark model has a

<sup>3</sup><http://www.statmt.org/europarl/>

<sup>4</sup><http://www.statmt.org/wmt0{6,7}>



Training	Decoding	
	derivation	translation
All Derivations	28.71	31.23
Single Derivation	26.70	27.32
ML ( $\sigma^2 = \infty$ )	25.57	25.97

**Table 1.** A comparison on the impact of accounting for all derivations in training and decoding (development set).

positive count for every rule (7.8M), the discriminative model only observes 1.7M rules in actual reference derivations. Figure 1 illustrates the massive ambiguity present in the training data, with fifteen word sentences averaging over 70M reference derivations.

Performance is evaluated using cased BLEU4 score on the test set. Although there is no direct relationship between BLEU and likelihood, it provides a rough measure for comparing performance.

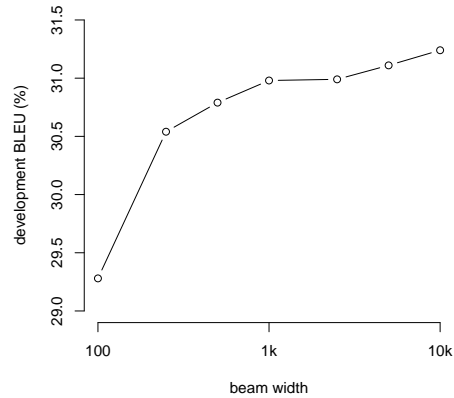
**Derivational ambiguity** Table 1 shows the impact of accounting for derivational ambiguity in training and decoding.<sup>5</sup> There are two options for training, we could use our latent variable model and optimise the probability of all derivations of the reference translation, or choose a single derivation that yields the reference and optimise its probability alone. The second option raises the difficult question of which one, of the thousands available, we should choose? We use the derivation which contains the most rules. The intuition is that small rules are likely to appear more frequently, and thus generalise better to a test set. In decoding we can search for the maximum probability derivation, which is the standard practice in SMT, or for the maximum probability translation which is what we actually want from our model, i.e. the best translation.

The results clearly indicate the value in optimising translations, rather than derivations. Max-translation decoding for the model trained on single derivations has only a small positive effect, while for the latent variable model the impact is much larger.<sup>6</sup>

For example, our max-derivation model trained on the Europarl data translates *carte sur la table* as *on the table card*. This error in the reordering of *card* (which is an acceptable translation of *carte*) is due to the rule  $\langle X \rangle \rightarrow \langle \text{carte } X_{\square}, X_{\square} \text{ card} \rangle$  being the highest scoring rule for *carte*. This is reasonable, as

<sup>5</sup>When not explicitly stated, both here and in subsequent results, the regularisation parameter was set to one,  $\sigma^2 = 1$ .

<sup>6</sup>We also experimented with using max-translation decoding for standard MER trained translation models, finding that it had a small negative impact on BLEU score.



**Figure 5.** The effect of the beam width (log-scale) on max-translation decoding (development set).

*carte* is a noun, which in the training data, is often observed with a trailing adjective which needs to be reordered when translating into English. In the example there is no adjective, but the simple hierarchical grammar cannot detect this. The max-translation model finds a good translation *card on the table*. This is due to the many rules that enforce monotone ordering around *sur la*,  $\langle X \rangle \rightarrow \langle X_{\square} \text{ sur}, X_{\square} \text{ in} \rangle$   $\langle X \rangle \rightarrow \langle X_{\square} \text{ sur la } X_{\square}, X_{\square} \text{ in the } X_{\square} \rangle$  etc. The scores of these many monotone rules sum to be greater than the reordering rule, thus allowing the model to use the weight of evidence to settle on the correct ordering.

Having established that the search for the best translation is effective, the question remains as to how the beam width over partial translations affects performance. Figure 5 shows the relationship between beam width and development BLEU. Even with a very tight beam of 100, max-translation decoding outperforms maximum-derivation decoding, and performance is increasing even at a width of 10k. In subsequent experiments we use a beam of 5k which provides a good trade-off between performance and speed.

**Regularisation** Table 1 shows that the performance of an unregularised maximum likelihood model lags well behind the regularised max-translation model. From this we can conclude that the maximum likelihood model is overfitting the training set. We suggest that is a result of the degenerate solutions of the conditional maximum likelihood estimate, as described in DeNero et al. (2006). Here we assert that our regularised *maximum a pos-*

Grammar Rules	ML ( $\sigma^2 = \infty$ )	MAP ( $\sigma^2 = 1$ )
$\langle X \rangle \rightarrow \langle \text{carte, map} \rangle$	1.0	0.5
$\langle X \rangle \rightarrow \langle \text{carte, notice} \rangle$	0.0	0.5
$\langle X \rangle \rightarrow \langle \text{sur, on} \rangle$	1.0	1.0
$\langle X \rangle \rightarrow \langle \text{la, the} \rangle$	1.0	1.0
$\langle X \rangle \rightarrow \langle \text{table, table} \rangle$	1.0	0.5
$\langle X \rangle \rightarrow \langle \text{table, chart} \rangle$	0.0	0.5
$\langle X \rangle \rightarrow \langle \text{carte sur, notice on} \rangle$	1.0	0.5
$\langle X \rangle \rightarrow \langle \text{carte sur, map on} \rangle$	0.0	0.5
$\langle X \rangle \rightarrow \langle \text{sur la, on the} \rangle$	1.0	1.0
$\langle X \rangle \rightarrow \langle \text{la table, the table} \rangle$	0.0	0.5
$\langle X \rangle \rightarrow \langle \text{la table, the chart} \rangle$	1.0	0.5
Training data: carte sur la table $\leftrightarrow$ map on the table carte sur la table $\leftrightarrow$ notice on the chart		

**Table 2.** Comparison of the susceptibility to degenerate solutions for a ML and MAP optimised model, using a simple grammar with one parameter per rule and a monotone glue rule:  $\langle X \rangle \rightarrow \langle X_{\boxed{1}} X_{\boxed{2}}, X_{\boxed{1}} X_{\boxed{2}} \rangle$

*teriori* model avoids such solutions.

This is illustrated in Table 2, which shows the conditional probabilities for rules, obtained by locally normalising the rule feature weights for a simple grammar extracted from the ambiguous pair of sentences presented in DeNero et al. (2006). The first column of conditional probabilities corresponds to a maximum likelihood estimate, i.e., without regularisation. As expected, the model finds a degenerate solution in which overlapping rules are exploited in order to minimise the entropy of the rule translation distributions. The second column shows the solution found by our model when regularised by a Gaussian prior with unit variance. Here we see that the model finds the desired solution in which the true ambiguity of the translation rules is preserved. The intuition is that in order to find a degenerate solution, dispreferred rules must be given large negative weights. However the prior penalises large weights, and therefore the best strategy for the regularised model is to evenly distribute probability mass.

**Translation comparison** Having demonstrated that accounting for derivational ambiguity leads to improvements for our discriminative model, we now place the performance of our system in the context of the standard approach to hierarchical translation. To do this we use our own implementation of Hiero (Chiang, 2007), with the same grammar but with the traditional generative feature set trained in a linear model with minimum BLEU training. The feature set includes: a trigram language model (*lm*) trained

System	Test (BLEU)
Discriminative max-derivation	25.78
Hiero ( $p_d, gr, rc, wc$ )	26.48
Discriminative max-translation	27.72
Hiero ( $p_d, p_r, p_d^{lex}, p_r^{lex}, gr, rc, wc$ )	28.14
Hiero ( $p_d, p_r, p_d^{lex}, p_r^{lex}, gr, rc, wc, lm$ )	32.00

**Table 3.** Test set performance compared with a standard Hiero system

on the English side of the unfiltered Europarl corpus; direct and reverse translation scores estimated as relative frequencies ( $p_d, p_r$ ); lexical translation scores ( $p_d^{lex}, p_r^{lex}$ ), a binary flag for the glue rule which allows the model to (dis)favour monotone translation (*gr*); and rule and target word counts (*rc, wc*).

Table 3 shows the results of our system on the test set. Firstly we show the relative scores of our model against Hiero without using reverse translation or lexical features.<sup>7</sup> This allows us to directly study the differences between the two translation models without the added complication of the other features. As well as both modelling the same distribution, when our model is trained with a single parameter per-rule these systems have the same parameter space, differing only in the manner of estimation.

Additionally we show the scores achieved by MERT training the full set of features for Hiero, with and without a language model.<sup>8</sup> We provide these results for reference. To compare our model directly with these systems we would need to incorporate additional features and a language model, work which we have left for a later date.

The relative scores confirm that our model, with its minimalist feature set, achieves comparable performance to the standard feature set without the language model. This is encouraging as our model was trained to optimise likelihood rather than BLEU, yet it is still competitive on that metric. As expected, the language model makes a significant difference to BLEU, however we believe that this effect is orthogonal to the choice of base translation model, thus we would expect a similar gain when integrating a language model into the discriminative system.

An informal comparison of the outputs on the development set, presented in Table 4, suggests that the

<sup>7</sup>Although the most direct comparison for the discriminative model would be with  $p_d$  model alone, omitting the *gr, rc* and *wc* features and MERT training produces poor translations.

<sup>8</sup>Hiero ( $p_d, p_r, p_d^{lex}, p_r^{lex}, gr, rc, wc, lm$ ) represents state-of-the-art performance on this training/testing set.

<b>S:</b> C'est pourquoi nous souhaitons que l'affaire nous soit renvoyée.
<b>R:</b> We therefore want the matter re-referred to ourselves.
<b>D:</b> That is why we want the that matters we to be referred back.
<b>T:</b> That is why we would like the matter to be referred back.
<b>H:</b> That is why we wish that the matter we be referred back.
<b>S:</b> Par contre, la transposition dans les États membres reste trop lente.
<b>R:</b> But implementation by the Member States has still been too slow.
<b>D:</b> However, it is implemented in the Member States is still too slow.
<b>T:</b> However, the implementation measures in Member States remains too slow.
<b>H:</b> In against, transposition in the Member States remains too slow.
<b>S:</b> Aussi, je considère qu'il reste énormément à faire dans ce domaine.
<b>R:</b> I therefore consider that there is an incredible amount still to do in this area.
<b>D:</b> So I think remains a lot to be done in this field.
<b>T:</b> So I think there is still much to be done in this area.
<b>H:</b> Therefore, I think it remains a vast amount to do in this area.

**Table 4.** Example output produced by the max-derivation (D), max-translation (T) decoding algorithms and Hiero( $p_d, p_r, p_d^{lex}, p_r^{lex}, gr, rc, wc$ ) (H) models, relative to the source (S) and reference (R).

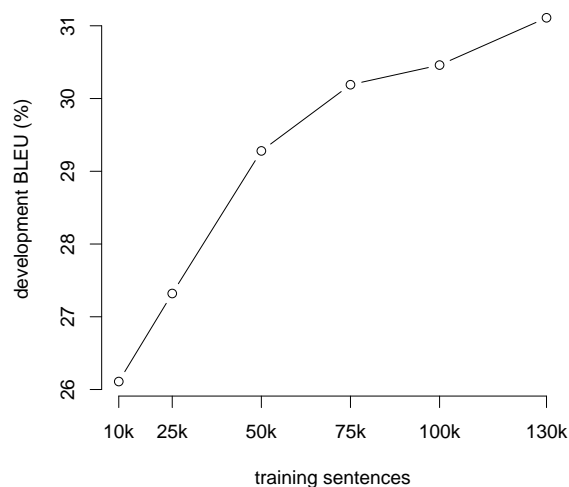
translation optimising discriminative model more often produces quite fluent translations, yet not in ways that would lead to an increase in BLEU score.<sup>9</sup> This could be considered a side-effect of optimising likelihood rather than BLEU.

**Scaling** In Figure 6 we plot the scaling characteristics of our models. The systems shown in the graph use the full grammar extracted on the 170k sentence corpus. The number of sentences upon which the iterative training algorithm is used to estimate the parameters is varied from 10k to the maximum 130K for which our model can reproduce the reference translation. As expected, the more data used to train the system, the better the performance. However, as the performance is still increasing significantly when all the parseable sentences are used, it is clear that the system's performance is suffering from the large number (40k) of sentences that are discarded before training.

## 5 Discussion and Further Work

We have shown that explicitly accounting for competing derivations yields translation improvements.

<sup>9</sup>Hiero was MERT trained on this set and has a 2% higher BLEU score compared to the discriminative model.



**Figure 6.** Learning curve showing that the model continues to improve as we increase the number of training sentences (development set)

Our model avoids the estimation biases associated with heuristic frequency count approaches and uses standard regularisation techniques to avoid degenerate maximum likelihood solutions.

Having demonstrated the efficacy of our model with very simple features, the logical next step is to investigate more expressive features. Promising features might include those over source side re-ordering rules (Wang et al., 2007) or source context features (Carpuat and Wu, 2007). Rule frequency features extracted from large training corpora would help the model to overcome the issue of unreachable reference sentences. Such approaches have been shown to be effective in log-linear word-alignment models where only a small supervised corpus is available (Blunsom and Cohn, 2006).

Finally, while in this paper we have focussed on the science of discriminative machine translation, we believe that with suitable engineering this model will advance the state-of-the-art. To do so would require integrating a language model feature into the max-translation decoding algorithm. The use of richer, more linguistic grammars (e.g., Galley et al. (2004)) may also improve the system.

## Acknowledgements

The authors acknowledge the support of the EPSRC (Blunsom & Osborne, grant EP/D074959/1; Cohn, grant GR/T04557/01).

## References

- Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proc. of the 44th Annual Meeting of the ACL and 21st International Conference on Computational Linguistics (COLING/ACL-2006)*, pages 65–72, Sydney, Australia, July.
- Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proc. of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-2007)*, pages 61–72, Prague, Czech Republic.
- David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of the 43rd Annual Meeting of the ACL (ACL-2005)*, pages 263–270, Ann Arbor, Michigan, June.
- David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
- Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proc. of the 42nd Annual Meeting of the ACL (ACL-2004)*, pages 103–110, Barcelona, Spain.
- Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).
- John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *Proc. of the HLT-NAACL 2006 Workshop on Statistical Machine Translation*, pages 31–38, New York City, June.
- Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What’s in a translation rule? In *Proc. of the 4th International Conference on Human Language Technology Research and 5th Annual Meeting of the NAACL (HLT-NAACL 2004)*, Boston, USA, May.
- Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of the 44th Annual Meeting of the ACL and 21st International Conference on Computational Linguistics (COLING/ACL-2006)*, pages 961–968, Sydney, Australia, July.
- Abraham Ittycheriah and Salim Roukos. 2007. Direct translation model 2. In *Proc. of the 7th International Conference on Human Language Technology Research and 8th Annual Meeting of the NAACL (HLT-NAACL 2007)*, pages 57–64, Rochester, USA.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of the 3rd International Conference on Human Language Technology Research and 4th Annual Meeting of the NAACL (HLT-NAACL 2003)*, pages 81–88, Edmonton, Canada, May.
- Philip M. Lewis II and Richard E. Stearns. 1968. Syntax-directed transduction. *J. ACM*, 15(3):465–488.
- Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of the 44th Annual Meeting of the ACL and 21st International Conference on Computational Linguistics (COLING/ACL-2006)*, pages 761–768, Sydney, Australia, July.
- Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. of the 6th Conference on Natural Language Learning (CoNLL-2002)*, pages 49–55, Taipei, Taiwan, August.
- Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the 41st Annual Meeting of the ACL (ACL-2003)*, pages 160–167, Sapporo, Japan.
- Slav Petrov, Adam Pauls, and Dan Klein. 2007. Discriminative log-linear grammars with latent variables. In *Advances in Neural Information Processing Systems 20 (NIPS)*, Vancouver, Canada.
- Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of the 3rd International Conference on Human Language Technology Research and 4th Annual Meeting of the NAACL (HLT-NAACL 2003)*, pages 134–141, Edmonton, Canada.
- Christoph Tillmann and Tong Zhang. 2007. A block bigram prediction model for statistical machine translation. *ACM Transactions Speech Language Processing*, 4(3):6.
- Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proc. of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-2007)*, pages 737–745, Prague, Czech Republic.
- Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proc. of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-2007)*, pages 764–773, Prague, Czech Republic.
- Benjamin Wellington, Joseph Turian, Chris Pike, and I. Dan Melamed. 2006. Scalable purely-discriminative training for word and tree transducers. In *Proc. of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, USA.
- Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

# Efficient Multi-pass Decoding for Synchronous Context Free Grammars

Hao Zhang and Daniel Gildea  
Computer Science Department  
University of Rochester  
Rochester, NY 14627

## Abstract

We take a multi-pass approach to machine translation decoding when using synchronous context-free grammars as the translation model and  $n$ -gram language models: the first pass uses a bigram language model, and the resulting parse forest is used in the second pass to guide search with a trigram language model. The trigram pass closes most of the performance gap between a bigram decoder and a much slower trigram decoder, but takes time that is insignificant in comparison to the bigram pass. An additional fast decoding pass maximizing the expected count of correct translation hypotheses increases the BLEU score significantly.

## 1 Introduction

Statistical machine translation systems based on synchronous grammars have recently shown great promise, but one stumbling block to their widespread adoption is that the decoding, or search, problem during translation is more computationally demanding than in phrase-based systems. This complexity arises from the interaction of the tree-based translation model with an  $n$ -gram language model. Use of longer  $n$ -grams improves translation results, but exacerbates this interaction. In this paper, we present three techniques for attacking this problem in order to obtain fast, high-quality decoders.

First, we present a two-pass decoding algorithm, in which the first pass explores states resulting from an integrated bigram language model, and the second pass expands these states into trigram-based

states. The general bigram-to-trigram technique is common in speech recognition (Murveit et al., 1993), where lattices from a bigram-based decoder are re-scored with a trigram language model. We examine the question of whether, given the reordering inherent in the machine translation problem, lower order  $n$ -grams will provide as valuable a search heuristic as they do for speech recognition.

Second, we explore heuristics for agenda-based search, and present a heuristic for our second pass that combines precomputed language model information with information derived from the first pass. With this heuristic, we achieve the same BLEU scores and model cost as a trigram decoder with essentially the same speed as a bigram decoder.

Third, given the significant speedup in the agenda-based trigram decoding pass, we can rescore the trigram forest to maximize the expected count of correct synchronous constituents of the model, using the product of inside and outside probabilities. Maximizing the expected count of synchronous constituents approximately maximizes BLEU. We find a significant increase in BLEU in the experiments, with minimal additional time.

## 2 Language Model Integrated Decoding for SCFG

We begin by introducing Synchronous Context Free Grammars and their decoding algorithms when an  $n$ -gram language model is integrated into the grammatical search space.

A **synchronous CFG** (SCFG) is a set of context-free rewriting rules for recursively generating string pairs. Each synchronous rule is a pair of CFG rules

with the nonterminals on the right hand side of one CFG rule being one-to-one mapped to the other CFG rule via a permutation  $\pi$ . We adopt the SCFG notation of Satta and Peserico (2005). Superscript *indices* in the right-hand side of grammar rules:

$$X \rightarrow X_1^{(1)} \dots X_n^{(n)}, X_{\pi(1)}^{(\pi(1))} \dots X_{\pi(n)}^{(\pi(n))}$$

indicate that the nonterminals with the same index are linked across the two languages, and will eventually be rewritten by the same rule application. Each  $X_i$  is a variable which can take the value of any non-terminal in the grammar.

In this paper, we focus on binary SCFGs and without loss of generality assume that only the pre-terminal unary rules can generate terminal string pairs. Thus, we are focusing on Inversion Transduction Grammars (Wu, 1997) which are an important subclass of SCFG. Formally, the rules in our grammar include preterminal unary rules:

$$X \rightarrow \mathbf{e/f}$$

for pairing up words or phrases in the two languages and binary production rules with straight or inverted orders that are responsible for building up upper-level synchronous structures. They are straight rules written:

$$X \rightarrow [YZ]$$

and inverted rules written:

$$X \rightarrow \langle YZ \rangle.$$

Most practical non-binary SCFGs can be binarized using the synchronous binarization technique by Zhang et al. (2006). The Hiero-style rules of (Chiang, 2005), which are not strictly binary but binary only on nonterminals:

$$X \rightarrow \text{yu } X^{(1)} \text{ you } X^{(2)}; \text{ have } X^{(2)} \text{ with } X^{(1)}$$

can be handled similarly through either offline binarization or allowing a fixed maximum number of gap words between the right hand side nonterminals in the decoder.

For these reasons, the parsing problems for more realistic synchronous CFGs such as in Chiang (2005) and Galley et al. (2006) are formally equivalent to ITG. Therefore, we believe our focus on ITG

for the search efficiency issue is likely to generalize to other SCFG-based methods.

Without an  $n$ -gram language model, decoding using SCFG is not much different from CFG parsing. At each time a CFG rule is applied on the input string, we apply the synchronized CFG rule for the output language. From a dynamic programming point of view, the DP states are  $X[i, j]$ , where  $X$  ranges over all possible nonterminals and  $i$  and  $j$  range over 0 to the input string length  $|w|$ . Each state stores the best translations obtainable. When we reach the top state  $S[0, |w|]$ , we can get the best translation for the entire sentence. The algorithm is  $O(|w|^3)$ .

However, when we want to integrate an  $n$ -gram language model into the search, our goal is searching for the derivation whose total sum of weights of productions and  $n$ -gram log probabilities is maximized. Now the adjacent span-parameterized states  $X[i, k]$  and  $X[k, j]$  can interact with each other by “peeping into” the leading and trailing  $n - 1$  words on the output side for each state. Different boundary words differentiate the span-parameterized states. Thus, to preserve the dynamic programming property, we need to refine the states by adding the boundary words into the parameterization. The *LM*-integrated states are represented as  $X[i, j, u_{1,\dots,n-1}, v_{1,\dots,n-1}]$ . Since the number of variables involved at each DP step has increased to  $3 + 4(n - 1)$ , the decoding algorithm is asymptotically  $O(|w|^{3+4(n-1)})$ . Although it is possible to use the “hook” trick of Huang et al. (2005) to factorize the DP operations to reduce the complexity to  $O(|w|^{3+3(n-1)})$ , when  $n$  is greater than 2, the complexity is still prohibitive.

### 3 Multi-pass LM-Integrated Decoding

In this section, we describe a multi-pass progressive decoding technique that gradually augments the *LM*-integrated states from lower orders to higher orders. For instance, a bigram-integrated state  $[X, i, j, u, v]$  is said to be a coarse-level state of a trigram-integrated state  $[X, i, j, u, u', v', v]$ , because the latter state refines the previous by specifying more inner words.

Progressive search has been used for HMM’s in speech recognition (Murveit et al., 1993). The gen-

eral idea is to use a simple and fast decoding algorithm to constrain the search space of a following more complex and slower technique. More specifically, a bigram decoding pass is executed forward and backward to figure out the probability of each state. Then the states can be pruned based on their global score using the product of inside and outside probabilities. The advanced decoding algorithm will use the constrained space (a lattice in the case of speech recognition) as a grammatical constraint to help it focus on a smaller search space on which more discriminative features are brought in.

The same idea has been applied to forests for parsing. Charniak and Johnson (2005) use a PCFG to do a pass of inside-outside parsing to reduce the state space of a subsequent lexicalized  $n$ -best parsing algorithm to produce parses that are further re-ranked by a MaxEnt model.

We take the same view as in speech recognition that a trigram integrated model is a finer-grained model than bigram model and in general we can do an  $n - 1$ -gram decoding as a predicative pass for the following  $n$ -gram pass. We need to do inside-outside parsing as coarse-to-fine parsers do. However, we use the outside probability or cost information differently. We do not combine the inside and outside costs of a simpler model to prune the space for a more complex model. Instead, for a given finer-grained state, we combine its true inside cost with the outside cost of its coarse-level counter-part to estimate its worthiness of being explored. The use of the outside cost from a coarser-level as the outside estimate makes our method naturally fall in the framework of A\* parsing.

Klein and Manning (2003) describe an A\* parsing framework for monolingual parsing and admissible outside estimates that are computed using inside/outside parsing algorithm on simplified PCFGs compared to the original PCFG. Zhang and Gildea (2006) describe A\* for ITG and develop admissible heuristics for both alignment and decoding. Both have shown the effectiveness of A\* in situations where the outside estimate approximates the true cost closely such as when the sentences are short. For decoding long sentences, it is difficult to come up with good admissible (or inadmissible) heuristics. If we can afford a bigram decoding pass, the outside cost from a bigram model is conceivably a

very good estimate of the outside cost using a trigram model since a bigram language model and a trigram language model must be strongly correlated. Although we lose the guarantee that the bigram-pass outside estimate is admissible, we expect that it approximates the outside cost very closely, thus very likely to effectively guide the heuristic search.

### 3.1 Inside-outside Coarse Level Decoding

We describe the coarse level decoding pass in this section. The decoding algorithms for the coarse level and the fine level do not necessarily have to be the same. The fine level decoding algorithm is an A\* algorithm. The coarse level decoding algorithm can be CKY or A\* or other alternatives.

Conceptually, the algorithm is finding the shortest hyperpath in the hypergraph in which the nodes are states like  $X[i, j, u_{1, \dots, n-1}, v_{1, \dots, n-1}]$ , and the hyperedges are the applications of the synchronous rules to go from right-hand side states to left-hand side states. The root of the hypergraph is a special node  $S'[0, |w|, \langle s \rangle, \langle /s \rangle]$  which means the entire input sentence has been translated to a string starting with the beginning-of-sentence symbol and ending at the end-of-sentence symbol. If we imagine a starting node that goes to all possible basic translation pairs, i.e., the instances of the terminal translation rules for the input, we are searching the shortest hyperpath from the imaginary bottom node to the root. To help our outside parsing pass, we store the back-pointers at each step of exploration.

The outside parsing pass, however, starts from the root  $S'[|w|, \langle s \rangle, \langle /s \rangle]$  and follows the back-pointers downward to the bottom nodes. The nodes need to be visited in a topological order so that whenever a node is visited, its parents have been visited and its outside cost is over all possible outside parses. The algorithm is described in pseudocode in Algorithm 1. The number of hyperedges to traverse is much fewer than in the inside pass because not every state explored in the bottom up inside pass can finally reach the goal. As for normal outside parsing, the operations are the reverse of inside parsing. We propagate the outside cost of the parent to its children by combining with the inside cost of the other children and the interaction cost, i.e., the language model cost between the focused child and the other children. Since we want to approximate the Viterbi

outside cost, it makes sense to maximize over all possible outside costs for a given node, to be consistent with the maximization of the inside pass. For the nodes that have been explored in the bottom up pass but not in the top-down pass, we set their outside cost to be infinity so that their exploration is preferred only when the viable nodes from the first pass have all been explored in the fine pass.

### 3.2 Heuristics for Fine-grained Decoding

In this section, we summarize the heuristics for finer level decoding.

The motivation for combining the true inside cost of the fine-grained model and the outside estimate given by the coarse-level parsing is to approximate the true global cost of a fine-grained state as closely as possible. We can make the approximation even closer by incorporating local higher-order outside  $n$ -gram information for a state of  $X[i, j, u_{1,\dots,n-1}, v_{1,\dots,n-1}]$  into account. We call this the *best-border* estimate. For example, the best-border estimate for trigram states is:

$$h_{BB}(X, i, j, u_1, u_2, v_1, v_2) = \left[ \max_{s \in S(i, j)} P_{lm}(u_2 | s, u_1) \right] \cdot \left[ \max_{s \in S(i, j)} P_{lm}(s | v_1, v_2) \right]$$

where  $S(i, j)$  is the set of candidate target language words outside the span of  $(i, j)$ .  $h_{BB}$  is the product of the upper bounds for the two on-the-border  $n$ -grams.

This heuristic function was one of the admissible heuristics used by Zhang and Gildea (2006). The benefit of including the best-border estimate is to refine the outside estimate with respect to the inner words which refine the bigram states into the trigram states. If we do not take the inner words into consideration when computing the outside cost, all states that map to the same coarse level state would have the same outside cost. When the simple best-border estimate is combined with the coarse-level outside estimate, it can further boost the search as will be shown in the experiments. To summarize, our recipe

for faster decoding is that using

$$\beta(X[i, j, u_{1,\dots,n-1}, v_{1,\dots,n-1}]) + \alpha(X[i, j, u_1, v_{n-1}]) + h_{BB}(X, i, j, u_{1,\dots,n}, v_{1,\dots,n}) \quad (1)$$

where  $\beta$  is the Viterbi inside cost and  $\alpha$  is the Viterbi outside cost, to globally prioritize the  $n$ -gram integrated states on the agenda for exploration.

### 3.3 Alternative Efficient Decoding Algorithms

The complexity of  $n$ -gram integrated decoding for SCFG has been tackled using other methods.

The hook trick of Huang et al. (2005) factorizes the dynamic programming steps and lowers the asymptotic complexity of the  $n$ -gram integrated decoding, but has not been implemented in large-scale systems where massive pruning is present.

The cube-pruning by Chiang (2007) and the lazy cube-pruning of Huang and Chiang (2007) turn the computation of beam pruning of CYK decoders into a top- $k$  selection problem given two columns of translation hypotheses that need to be combined. The insight for doing the expansion top-down lazily is that there is no need to uniformly explore every cell. The algorithm starts with requesting the first best hypothesis from the root. The request translates into requests for the  $k$ -bests of some of its children and grandchildren and so on, because re-ranking at each node is needed to get the top ones.

Venugopal et al. (2007) also take a two-pass decoding approach, with the first pass leaving the language model boundary words out of the dynamic programming state, such that only one hypothesis is retained for each span and grammar symbol.

## 4 Decoding to Maximize BLEU

The ultimate goal of efficient decoding to find the translation that has a highest evaluation score using the least time possible. Section 3 talks about utilizing the outside cost of a lower-order model to estimate the outside cost of a higher-order model, boosting the search for the higher-order model. By doing so, we hope the intrinsic metric of our model agrees with the extrinsic metric of evaluation so that fast search for the model is equivalent to efficient decoding. But the mismatch between the two is evident, as we will see in the experiments. In this section,



---

**Algorithm 1** OutsideCoarseParsing()

---

```
for all  $X[i, j, u, v]$  in topological order do
  for all children pairs pointed to by the back-pointers do
    if  $X \rightarrow [YZ]$  then
       $\triangleright$  the two children are  $Y[i, k, u, u']$  and  $Z[k, j, v', v]$ 
       $\alpha(Y[i, k, u, u']) = \max \{ \alpha(Y[i, k, u, u']),$ 
         $\alpha(X[i, j, u, v]) + \beta(Z[k, j, v', v]) + rule(X \rightarrow [YZ]) + bigram(u', v') \}$ 
       $\alpha(Z[k, j, v', v]) = \max \{ \alpha(Z[k, j, v', v]),$ 
         $\alpha(X[i, j, u, v]) + \beta(Y[i, k, u, u']) + rule(X \rightarrow [YZ]) + bigram(u', v') \}$ 
    end if
    if  $X \rightarrow \langle YZ \rangle$  then
       $\triangleright$  the two children are  $Y[i, k, v', v]$  and  $Z[k, j, u, u']$ 
       $\alpha(Y[i, k, v', v]) = \max \{ \alpha(Y[i, k, v', v]),$ 
         $\alpha(X[i, j, u, v]) + \beta(Z[k, j, u, u']) + rule(X \rightarrow \langle YZ \rangle) + bigram(u', v') \}$ 
       $\alpha(Z[k, j, u, u']) = \max \{ \alpha(Z[k, j, u, u']),$ 
         $\alpha(X[i, j, u, v]) + \beta(Y[i, k, v', v]) + rule(X \rightarrow \langle YZ \rangle) + bigram(u', v') \}$ 
    end if
  end for
end for
```

---

we deal with the mismatch by introducing another decoding pass that maximizes the expected count of synchronous constituents in the tree corresponding to the translation returned. BLEU is based on  $n$ -gram precision, and since each synchronous constituent in the tree adds a new 4-gram to the translation at the point where its children are concatenated, the additional pass approximately maximizes BLEU.

Kumar and Byrne (2004) proposed the framework of Minimum Bayesian Risk (MBR) decoding that minimizes the expected loss given a loss function. Their MBR decoding is a reranking pass over an  $n$ -best list of translations returned by the decoder. Our algorithm is another dynamic programming decoding pass on the trigram forest, and is similar to the parsing algorithm for maximizing expected labelled recall presented by Goodman (1996).

#### 4.1 Maximizing the expected count of correct synchronous constituents

We introduce an algorithm that maximizes the expected count of correct synchronous constituents. Given a synchronous constituent specified by the state  $[X, i, j, u, u', v', v]$ , its probability of being correct in the model is

$$\begin{aligned} EC([X, i, j, u, u', v', v]) \\ = \alpha([X, i, j, u, u', v', v]) \cdot \beta([X, i, j, u, u', v', v]), \end{aligned}$$

where  $\alpha$  is the outside probability and  $\beta$  is the inside probability. We approximate  $\beta$  and  $\alpha$  using the Viterbi probabilities. Since decoding from bottom up in the trigram pass already gives us the inside Viterbi scores, we only have to visit the nodes in the reverse order once we reach the root to compute the Viterbi outside scores. The outside-pass Algorithm 1 for bigram decoding can be generalized to the trigram case. We want to maximize over all translations (synchronous trees)  $T$  in the forest after the trigram decoding pass according to

$$\max_T \sum_{[X, i, j, u, u', v', v] \in T} EC([X, i, j, u, u', v', v]).$$

The expression can be factorized and computed using dynamic programming on the forest.

## 5 Experiments

We did our decoding experiments on the LDC 2002 MT evaluation data set for translation of Chinese newswire sentences into English. The evaluation data set has 10 human translation references for each sentence. There are a total of 371 Chinese sentences of no more than 20 words in the data set. These sentences are the test set for our different versions of language-model-integrated ITG decoders. We evaluate the translation results by comparing them against the reference translations using the BLEU metric.

The word-to-word translation probabilities are from the translation model of IBM Model 4 trained on a 160-million-word English-Chinese parallel corpus using GIZA++. The phrase-to-phrase translation probabilities are trained on 833K parallel sentences. 758K of this was data made available by ISI, and another 75K was FBIS data. The language model is trained on a 30-million-word English corpus. The rule probabilities for ITG are trained using EM on a corpus of 18,773 sentence pairs with a total of 276,113 Chinese words and 315,415 English words.

### 5.1 Bigram-pass Outside Cost as Trigram-pass Outside Estimate

We first fix the beam for the bigram pass, and change the outside heuristics for the trigram pass to show the difference before and after using the first-pass outside cost estimate and the border estimate. We choose the beam size for the CYK bigram pass to be 10 on the log scale. The first row of Table 1 shows the number of explored hyperedges for the bigram pass and its BLEU score. In the rows below, we compare the additional numbers of hyperedges that need to be explored in the trigram pass using different outside heuristics. It takes too long to finish using uniform outside estimate; we have to use a tight beam to control the agenda-based exploration. Using the bigram outside cost estimate makes a huge difference. Furthermore, using Equation 1, adding the additional heuristics on the best trigrams that can appear on the borders of the current hypothesis, on average we only need to explore 2700 additional hyperedges per sentence to boost the BLEU score from 21.77 to 23.46. The boost is so significant that overall the dominant part of search time is no longer the second pass but the first bigram pass (inside pass actually) which provides a constrained space and outside heuristics for the second pass.

### 5.2 Two-pass decoding versus One-pass decoding

By varying the beam size for the first pass, we can plot graphs of model scores versus search time and BLEU scores versus search time as shown in Figure 1. We use a very large beam for the second pass due to the reason that the outside estimate for the second pass is discriminative enough to guide the

<i>Decoding Method</i>	Avg. Hyperedges	BLEU
Bigram Pass	167K	21.77
Trigram Pass		
UNI	–	–
BO	+ 629.7K=796.7K	23.56
<i>BO+BB</i>	+2.7K =169.7K	23.46
Trigram One-pass, with Beam	6401K	23.47

Table 1: Speed and BLEU scores for two-pass decoding. UNI stands for the uniform (zero) outside estimate. BO stands for the bigram outside cost estimate. BB stands for the best border estimate, which is added to BO.

<i>Decoder</i>	Time	BLEU	Model Score
One-pass agenda	4317s	22.25	-208.849
One-pass CYK	3793s	22.89	-207.309
<i>Multi-pass, CYK first</i>			
<i>agenda second pass</i>	3689s	23.56	-205.344
<b>MEC third pass</b>	<b>3749s</b>	<b>24.07</b>	<b>-203.878</b>
Lazy-cube-pruning	3746s	22.16	-208.575

Table 2: Summary of different trigram decoding strategies, using about the same time (10 seconds per sentence).

search. We sum up the total number of seconds for both passes to compare with the baseline systems. On average, less than 5% of time is spent in the second pass.

In Figure 1, we have four competing decoders. *bitri\_cyk* is our two-pass decoder, using CYK as the first pass decoding algorithm and using agenda-based decoding in the second pass which is guided by the first pass. *agenda* is our trigram-integrated agenda-based decoder. The other two systems are also one-pass. *cyk* is our trigram-integrated CYK decoder. *lazy\_kbest* is our top-down k-best-style decoder.<sup>1</sup>

Figure 1(left) compares the search efficiencies of the four systems. *bitri\_cyk* at the top ranks first. *cyk* follows it. The curves of *lazy\_kbest* and *agenda* cross

<sup>1</sup>In our implementation of the lazy-cube-pruning based ITG decoder, we vary the re-ranking buffer size and the top-*k* list size which are the two controlling parameters for the search space. But we did not use any *LM* estimate to achieve early stopping as suggested by Huang and Chiang (2007). Also, we did not have a translation-model-only pruning pass. So the results shown in this paper for the lazy cube pruning method is not of its best performance.

and are both below the curves of *bitri\_cyk* and *cyk*. This figure indicates the advantage of the two-pass decoding strategy in producing translations with a high model score in less time.

However, model scores do not directly translate into BLEU scores. In Figure 1(right), *bitri\_cyk* is better than *CYK* only in a certain time window when the beam is neither too small nor too large. But the window is actually where we are interested – it ranges from 5 seconds per sentence to 20 seconds per sentence. Table 2 summarizes the performance of the four decoders when the decoding speed is at 10 seconds per sentence.

### 5.3 Does the hook trick help?

We have many choices in implementing the bigram decoding pass. We can do either *CYK* or agenda-based decoding. We can also use the dynamic programming hook trick. We are particularly interested in the effect of the hook trick in a large-scale system with aggressive pruning.

Figure 2 compares the four possible combinations of the decoding choices for the first pass: *bitri\_cyk*, *bitri\_agenda*, *bitri\_cyk\_hook* and *bitri\_agenda\_hook*. *bitri\_cyk* which simply uses *CYK* as the first pass decoding algorithm is the best in terms of performance and time trade-off. The hook-based decoders do not show an advantage in our experiments. Only *bitri\_agenda\_hook* gets slightly better than *bitri\_agenda* when the beam size increases. So, it is very likely the overhead of building hooks offsets its benefit when we massively prune the hypotheses.

### 5.4 Maximizing BLEU

The *bitri\_cyk* decoder spends little time in the agenda-based trigram pass, quickly reaching the goal item starting from the bottom of the chart. In order to maximize BLEU score using the algorithm described in Section 4, we need a sizable trigram forest as a starting point. Therefore, we keep popping off more items from the agenda after the goal is reached. Simply by exploring more (200 times the log beam) after-goal items, we can optimize the Viterbi synchronous parse significantly, shown in Figure 3(left) in terms of model score versus search time.

However, the mismatch between model score and BLEU score persists. So, we try our algorithm

of maximizing expected count of synchronous constituents on the trigram forest. We find significant improvement in BLEU, as shown in Figure 3 (right) by the curve of *bitri\_cyk\_epass\_me\_cons*. *bitri\_cyk\_epass\_me\_cons* beats both *bitri\_cyk* and *cyk* in terms of BLEU versus time if using more than 1.5 seconds on average to decode each sentence. At each time point, the difference in BLEU between *bitri\_cyk\_epass\_me\_cons* and the highest of *bitri\_cyk* and *cyk* is around .5 points consistently as we vary the beam size for the first pass. We achieve the record-high BLEU score 24.34 using on average 21 seconds per sentence, compared to the next-highest score of 23.92 achieved by *cyk* using on average 78 seconds per sentence.

## 6 Conclusion

We present a multi-pass method to speed up  $n$ -gram integrated decoding for SCFG. We use an inside/outside parsing algorithm to get the Viterbi outside cost of bigram integrated states which is used as an outside estimate for trigram integrated states. The coarse-level outside cost plus the simple estimate for border trigrams speeds up the trigram decoding pass hundreds of times compared to using no outside estimate.

Maximizing the probability of the synchronous derivation is not equivalent to maximizing BLEU. We use a rescoring decoding pass that maximizes the expected count of synchronous constituents. This technique, together with the progressive search at previous stages, gives a decoder that produces the highest BLEU score we have obtained on the data in a very reasonable amount of time.

As future work, new metrics for the final pass may be able to better approximate BLEU. As the bigram decoding pass currently takes the bulk of the decoding time, better heuristics for this phase may speed up the system further.

**Acknowledgments** This work was supported by NSF ITR-0428020 and NSF IIS-0546554.

## References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine  $n$ -best parsing and maxent discriminative reranking. In *ACL*.

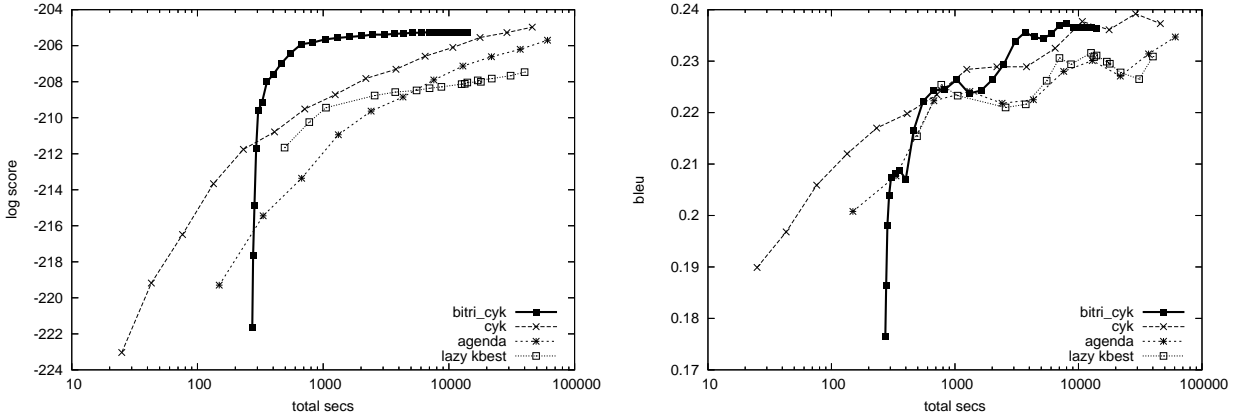


Figure 1: We compare the two-pass ITG decoder with the one-pass trigram-integrated ITG decoders in terms of both model scores vs. time (left) and BLEU scores vs. time (right). The model score here is the log probability of the decoded parse, summing up both the translation model and the language model. We vary the beam size (for the first pass in the case of two-pass) to search more and more thoroughly.

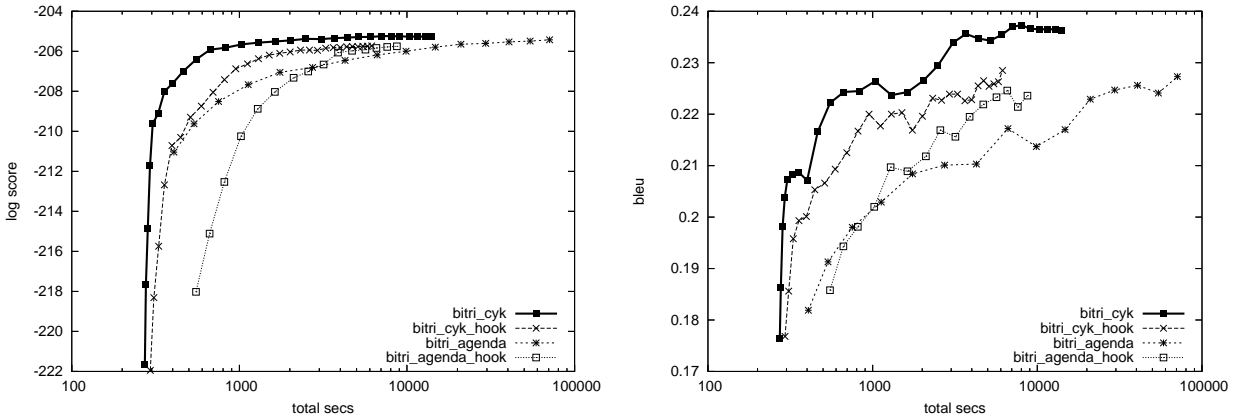


Figure 2: We use different first-pass decoding algorithms, fixing the second pass to be agenda-based which is guided by the outside cost of the first pass. Left: model score vs. time. Right: BLEU score vs. time.

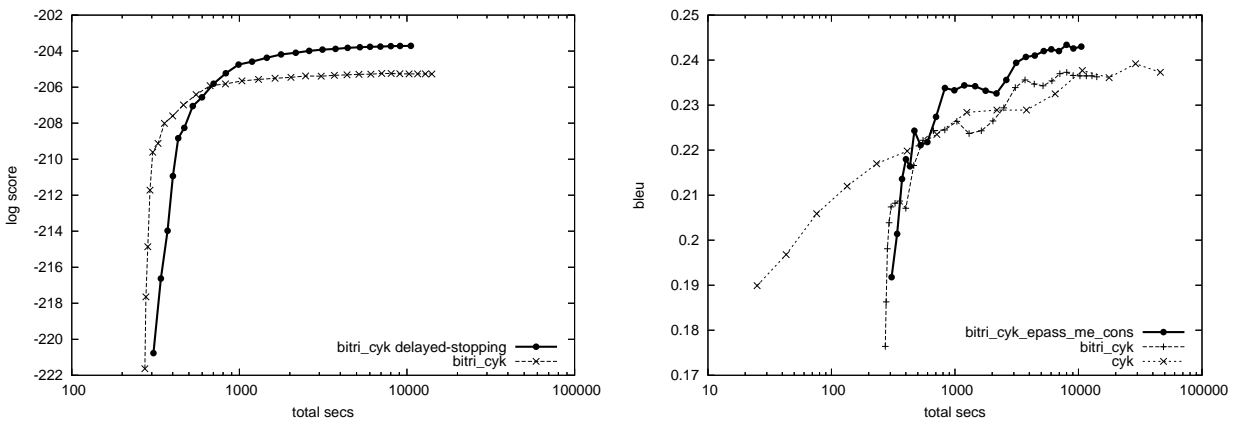


Figure 3: Left: improving the model score by extended agenda-exploration after the goal is reached in the best-first search. Right: maximizing BLEU by the maximizing expectation pass on the expanded forest.

- David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, pages 263–270.
- David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).
- Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the International Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06)*, pages 961–968, July.
- Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the 34th Annual Conference of the Association for Computational Linguistics (ACL-96)*, pages 177–183.
- Liang Huang and David Chiang. 2007. Faster algorithms for decoding with integrated language models. In *Proceedings of ACL*, Prague, June.
- Liang Huang, Hao Zhang, and Daniel Gildea. 2005. Machine translation as lexicalized parsing with hooks. In *International Workshop on Parsing Technologies (IWPT05)*, Vancouver, BC.
- Dan Klein and Christopher D. Manning. 2003. A\* parsing: Fast exact Viterbi parse selection. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*.
- Shankar Kumar and William Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 169–176, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.
- Hy Murveit, John W. Butzberger, Vassilios V. Digalakis, and Mitchel Weintraub. 1993. Large-vocabulary dictation using SRI's decipher speech recognition system: Progressive-search techniques. In *Proceedings of the IEEE International Conference on Acoustics, Speech, & Signal Processing (IEEE ICASSP-93)*, volume 2, pages 319–322. IEEE.
- Giorgio Satta and Enoch Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 803–810, Vancouver, Canada, October.
- Ashish Venugopal, Andreas Zollmann, and Stephan Vogel. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *NAACL07*, Rochester, NY, April.
- Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.
- Hao Zhang and Daniel Gildea. 2006. Efficient search for inversion transduction grammar. In *2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sydney.
- Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the 2006 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-06)*, pages 256–263.

# Regular tree grammars as a formalism for scope underspecification

Alexander Koller\*

a.koller@ed.ac.uk

\* University of Edinburgh

Michaela Regneri<sup>† §</sup>

regneri@coli.uni-sb.de

<sup>†</sup> University of Groningen

Stefan Thater<sup>§</sup>

stth@coli.uni-sb.de

<sup>§</sup> Saarland University

## Abstract

We propose the use of regular tree grammars (RTGs) as a formalism for the underspecified processing of scope ambiguities. By applying standard results on RTGs, we obtain a novel algorithm for eliminating equivalent readings and the first efficient algorithm for computing the best reading of a scope ambiguity. We also show how to derive RTGs from more traditional underspecified descriptions.

## 1 Introduction

Underspecification (Reyle, 1993; Copestake et al., 2005; Bos, 1996; Egg et al., 2001) has become the standard approach to dealing with scope ambiguity in large-scale hand-written grammars (see e.g. Copestake and Flickinger (2000)). The key idea behind underspecification is that the parser avoids computing all scope readings. Instead, it computes a single compact *underspecified description* for each parse. One can then strengthen the underspecified description to efficiently eliminate subsets of readings that were not intended in the given context (Koller and Niehren, 2000; Koller and Thater, 2006); so when the individual readings are eventually computed, the number of remaining readings is much smaller and much closer to the actual perceived ambiguity of the sentence.

In the past few years, a “standard model” of scope underspecification has emerged: A range of formalisms from Underspecified DRT (Reyle, 1993) to dominance graphs (Althaus et al., 2003) have offered mechanisms to specify the “semantic material” of which the semantic representations are built up, plus dominance or outscoping relations between these building blocks. This has been a very successful approach, but recent algorithms for eliminating subsets of readings have pushed the expres-

sive power of these formalisms to their limits; for instance, Koller and Thater (2006) speculate that further improvements over their (incomplete) redundancy elimination algorithm require a more expressive formalism than dominance graphs. On the theoretical side, Ebert (2005) has shown that none of the major underspecification formalisms are *expressively complete*, i.e. supports the description of an arbitrary subset of readings. Furthermore, the somewhat implicit nature of dominance-based descriptions makes it difficult to systematically associate readings with probabilities or costs and then compute a best reading.

In this paper, we address both of these shortcomings by proposing *regular tree grammars (RTGs)* as a novel underspecification formalism. Regular tree grammars (Comon et al., 2007) are a standard approach for specifying sets of trees in theoretical computer science, and are closely related to regular tree transducers as used e.g. in recent work on statistical MT (Knight and Graehl, 2005) and grammar formalisms (Shieber, 2006). We show that the “dominance charts” proposed by Koller and Thater (2005b) can be naturally seen as regular tree grammars; using their algorithm, classical underspecified descriptions (dominance graphs) can be translated into RTGs that describe the same sets of readings. However, RTGs are trivially expressively complete because every finite tree language is also regular. We exploit this increase in expressive power in presenting a novel redundancy elimination algorithm that is simpler and more powerful than the one by Koller and Thater (2006); in our algorithm, redundancy elimination amounts to intersection of regular tree languages. Furthermore, we show how to define a PCFG-style cost model on RTGs and compute best readings of deterministic RTGs efficiently, and illustrate this model on a machine learning based model

of scope preferences (Higgins and Sadock, 2003). To our knowledge, this is the first efficient algorithm for computing best readings of a scope ambiguity in the literature.

The paper is structured as follows. In Section 2, we will first sketch the existing standard approach to underspecification. We will then define regular tree grammars and show how to see them as an underspecification formalism in Section 3. We will present the new redundancy elimination algorithm, based on language intersection, in Section 4, and show how to equip RTGs with weights and compute best readings in Section 5. We conclude in Section 6.

## 2 Underspecification

The key idea behind scope underspecification is to describe all readings of an ambiguous expression with a single, compact underspecified representation (USR). This simplifies semantics construction, and current algorithms (Koller and Thater, 2005a) support the efficient enumeration of readings from an USR when it is necessary. Furthermore, it is possible to perform certain semantic processing tasks such as eliminating redundant readings (see Section 4) directly on the level of underspecified representations without explicitly enumerating individual readings.

Under the “standard model” of scope underspecification, readings are considered as formulas or trees. USRs specify the “semantic material” common to all readings, plus dominance or outscopes relations between these building blocks. In this paper, we consider dominance graphs (Egg et al., 2001; Althaus et al., 2003) as one representative of this class. An example dominance graph is shown on the left of Fig. 1. It represents the five readings of the sentence “a representative of a company saw every sample.” The (directed, labelled) graph consists of seven subtrees, or *fragments*, plus *dominance edges* relating nodes of these fragments. Each reading is encoded as one *configuration* of the dominance graph, which can be obtained by “plugging” the tree fragments into each other, in a way that respects the dominance edges: The source node of each dominance edge must dominate (i.e., be an ancestor of) the target node in each configuration. The trees in Fig. 1a–e are the five configurations of the example graph.

An important class of dominance graphs are *hy-*

*pernormally connected* dominance graphs, or *dominance nets* (Niehren and Thater, 2003). The precise definition of dominance nets is not important here, but note that virtually all underspecified descriptions that are produced by current grammars are nets (Flickinger et al., 2005). For the rest of the paper, we restrict ourselves to dominance graphs that are hypernormally connected.

## 3 Regular tree grammars

We will now recall the definition of regular tree grammars and show how they can be used as an underspecification formalism.

### 3.1 Definition

Let  $\Sigma$  be an alphabet, or signature, of tree constructors  $\{f, g, a, \dots\}$ , each of which is equipped with an arity  $\text{ar}(f) \geq 0$ . A *finite constructor tree*  $t$  is a finite tree in which each node is labelled with a symbol of  $\Sigma$ , and the number of children of the node is exactly the arity of this symbol. For instance, the configurations in Fig. 1a–e are finite constructor trees over the signature  $\{a_x|2, a_y|2, \text{comp}_z|0, \dots\}$ . Finite constructor trees can be seen as ground terms over  $\Sigma$  that respect the arities. We write  $T(\Sigma)$  for the finite constructor trees over  $\Sigma$ .

A *regular tree grammar* (RTG) is a 4-tuple  $G = (S, N, \Sigma, R)$  consisting of a *nonterminal alphabet*  $N$ , a *terminal alphabet*  $\Sigma$ , a *start symbol*  $S \in N$ , and a finite set of *production rules*  $R$  of the form  $A \rightarrow \beta$ , where  $A \in N$  and  $\beta \in T(\Sigma \cup N)$ ; the nonterminals count as zero-place constructors. Two finite constructor trees  $t, t' \in T(\Sigma \cup N)$  stand in the *derivation relation*,  $t \rightarrow_G t'$ , if  $t'$  can be built from  $t$  by replacing an occurrence of some nonterminal  $A$  by the tree on the right-hand side of some production for  $A$ . The *language generated by*  $G$ ,  $L(G)$ , is the set  $\{t \in T(\Sigma) \mid S \rightarrow_G^* t\}$ , i.e. all terms of terminal symbols that can be derived from the start symbol by a sequence of rule applications. Note that  $L(G)$  is a possibly infinite language of finite trees. As usual, we write  $A \rightarrow t_1 \mid \dots \mid t_n$  as shorthand for the  $n$  production rules  $A \rightarrow t_i$  ( $1 \leq i \leq n$ ). See Comon et al. (2007) for more details.

The languages that can be accepted by regular tree grammars are called *regular tree languages* (RTLs), and regular tree grammars are equivalent to *regular*

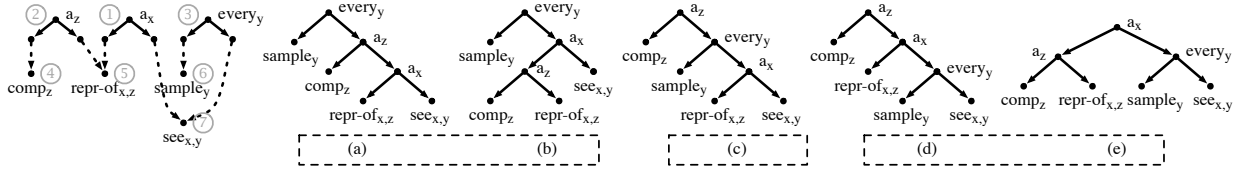


Figure 1: A dominance graph (left) and its five configurations.

*tree automata*, which are defined essentially like the well-known regular string automata, except that they assign states to the nodes in a tree rather than the positions in a string. Tree automata are related to tree transducers as used e.g. in statistical machine translation (Knight and Graehl, 2005) exactly like finite-state string automata are related to finite-state string transducers, i.e. they use identical mechanisms to accept rather than transduce languages. Many theoretical results carry over from regular string languages to regular tree languages; for instance, membership of a tree in a RTL can be decided in linear time, RTLs are closed under intersection, union, and complement, and so forth.

### 3.2 Regular tree grammars in underspecification

We can now use regular tree grammars in underspecification by representing the semantic representations as trees and taking an RTG  $G$  as an underspecified description of the trees in  $L(G)$ . For example, the five configurations in Fig. 1 can be represented as the tree language accepted by the following grammar with start symbol  $S$ .

$$\begin{aligned}
S &\rightarrow a_x(A_1, A_2) \mid a_z(B_1, A_3) \mid every_y(B_3, A_4) \\
A_1 &\rightarrow a_z(B_1, B_2) \\
A_2 &\rightarrow every_y(B_3, B_4) \\
A_3 &\rightarrow a_x(B_2, A_2) \mid every_y(B_3, A_5) \\
A_4 &\rightarrow a_x(A_1, B_4) \mid a_z(B_1, A_5) \\
A_5 &\rightarrow a_x(B_2, B_4) \\
B_1 &\rightarrow comp_z \quad B_2 \rightarrow repr-of_{x,z} \\
B_3 &\rightarrow sample_y \quad B_4 \rightarrow see_{x,y}
\end{aligned}$$

More generally, every finite set of trees can be written as the tree language accepted by a non-recursive regular tree grammar such as this. This grammar can be much smaller than the set of trees, because nonterminal symbols (which stand for sets of possibly many subtrees) can be used on the right-hand sides of multiple rules. Thus an RTG is a compact representation of a set of trees in the same way that a parse chart is a compact representation of the

set of parse trees of a context-free string grammar. Note that each tree can be enumerated from the RTG in linear time.

### 3.3 From dominance graphs to tree grammars

Furthermore, regular tree grammars can be systematically computed from more traditional underspecified descriptions. Koller and Thater (2005b) demonstrate how to compute a *dominance chart* from a dominance graph  $D$  by tabulating how a subgraph can be decomposed into smaller subgraphs by removing what they call a “free fragment”. If  $D$  is hypernormally connected, this chart can be read as a regular tree grammar whose nonterminal symbols are subgraphs of the dominance graph, and whose terminal symbols are names of fragments. For the example graph in Fig. 1, it looks as follows.

$$\begin{aligned}
\{1, 2, 3, 4, 5, 6, 7\} &\rightarrow 1(\{2, 4, 5\}, \{3, 6, 7\}) \\
\{1, 2, 3, 4, 5, 6, 7\} &\rightarrow 2(\{4\}, \{1, 3, 5, 6, 7\}) \\
\{1, 2, 3, 4, 5, 6, 7\} &\rightarrow 3(\{6\}, \{1, 2, 4, 5, 7\}) \\
\{1, 3, 5, 6, 7\} &\rightarrow 1(\{5\}, \{3, 6, 7\}) \mid 3(\{6\}, \{1, 5, 7\}) \\
\{1, 2, 4, 5, 7\} &\rightarrow 1(\{2, 4, 5\}, \{7\}) \mid 2(\{4\}, \{1, 5, 7\}) \\
\{1, 5, 7\} &\rightarrow 1(\{5\}, \{7\}) \\
\{2, 4, 5\} &\rightarrow 2(\{4\}, \{5\}) \quad \{4\} \rightarrow 4 \quad \{6\} \rightarrow 6 \\
\{3, 6, 7\} &\rightarrow 3(\{6\}, \{7\}) \quad \{5\} \rightarrow 5 \quad \{7\} \rightarrow 7
\end{aligned}$$

This grammar accepts, again, five different trees, whose labels are the node names of the dominance graph, for instance  $1(2(4, 5), 3(6, 7))$ . If  $f: \Sigma \rightarrow \Sigma'$  is a *relabelling function* from one terminal alphabet to another, we can write  $f(G)$  for the grammar  $(S, N, \Sigma', R')$ , where  $R' = \{A \rightarrow f(a)(B_1, \dots, B_n) \mid A \rightarrow a(B_1, \dots, B_n) \in R\}$ . Now if we choose  $f$  to be the labelling function of  $D$  (which maps node names to node labels) and  $G$  is the chart of  $D$ , then  $L(f(G))$  will be the set of configurations of  $D$ . The grammar in Section 3.2 is simply  $f(G)$  for the chart above (up to consistent renaming of nonterminals).

In the worst case, the dominance chart of a dominance graph with  $n$  fragments has  $O(2^n)$  production rules (Koller and Thater, 2005b), i.e. charts may be exponential in size; but note that this is still an



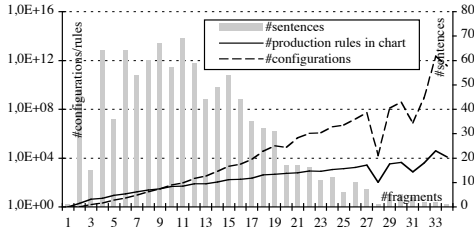


Figure 2: Chart sizes in the Rondane corpus.

improvement over the  $n!$  configurations that these worst-case examples have. In practice, RTGs that are computed by converting the USR computed by a grammar remain compact: Fig. 2 compares the average number of configurations and the average number of RTG production rules for USRs of increasing sizes in the Rondane treebank (see Sect. 4.3); the bars represent the number of sentences for USRs of a certain size. Even for the most ambiguous sentence, which has about  $4.5 \times 10^{12}$  scope readings, the dominance chart has only about 75 000 rules, and it takes only 15 seconds on a modern consumer PC (Intel Core 2 Duo at 2 GHz) to compute the grammar from the graph. Computing the charts for all 999 MRS-nets in the treebank takes about 45 seconds.

## 4 Expressive completeness and redundancy elimination

Because every finite tree language is regular, RTGs constitute an *expressively complete* underspecification formalism in the sense of Ebert (2005): They can represent arbitrary subsets of the original set of readings. Ebert shows that the classical dominance-based underspecification formalisms, such as MRS, Hole Semantics, and dominance graphs, are all expressively incomplete, which Koller and Thater (2006) speculate might be a practical problem for algorithms that strengthen USRs to remove unwanted readings. We will now show how both the expressive completeness and the availability of standard constructions for RTGs can be exploited to get an improved redundancy elimination algorithm.

### 4.1 Redundancy elimination

Redundancy elimination (Vestre, 1991; Chaves, 2003; Koller and Thater, 2006) is the problem of deriving from an USR  $U$  another USR  $U'$ , such that the readings of  $U'$  are a proper subset of the read-

ings of  $U$ , but every reading in  $U$  is semantically equivalent to some reading in  $U'$ . For instance, the following sentence from the Rondane treebank is analyzed as having six quantifiers and 480 readings by the ERG grammar; these readings fall into just two semantic equivalence classes, characterized by the relative scope of “the lee of” and “a small hillside”. A redundancy elimination would therefore ideally reduce the underspecified description to one that has only two readings (one for each class).

- (1) We quickly put up the tents in the lee of a small hillside and cook for the first time in the open. (Rondane 892)

Koller and Thater (2006) define semantic equivalence in terms of a rewrite system that specifies under what conditions two quantifiers may exchange their positions without changing the meaning of the semantic representation. For example, if we assume the following rewrite system (with just a single rule), the five configurations in Fig. 1a-e fall into three equivalence classes – indicated by the dotted boxes around the names a-e – because two pairs of readings can be rewritten into each other.

- (2)  $a_x(a_z(P, Q), R) \rightarrow a_z(P, a_x(Q, R))$

Based on this definition, Koller and Thater (2006) present an algorithm (henceforth, KT06) that deletes rules from a dominance chart and thus removes subsets of readings from the USR. The KT06 algorithm is fast and quite effective in practice. However, it essentially predicts for each production rule of a dominance chart whether each configuration that can be built with this rule is equivalent to a configuration that can be built with some other production for the same subgraph, and is therefore rather complex.

### 4.2 Redundancy elimination as language intersection

We now define a new algorithm for redundancy elimination. It is based on the intersection of regular tree languages, and will be much simpler and more powerful than KT06.

Let  $G = (S, N, \Sigma, R)$  be an RTG with a linear order on the terminals  $\Sigma$ ; for ease of presentation, we assume  $\Sigma \subseteq \mathbb{N}$ . Furthermore, let  $f : \Sigma \rightarrow \Sigma'$  be a relabelling function into the signature  $\Sigma'$  of the rewrite

system. For example,  $G$  could be the dominance chart of some dominance graph  $D$ , and  $f$  could be the labelling function of  $D$ .

We can then define a tree language  $L_F$  as follows:  $L_F$  contains all trees over  $\Sigma$  that do not contain a subtree of the form  $q_1(x_1, \dots, x_{i-1}, q_2(\dots), x_{i+1}, \dots, x_k)$  where  $q_1 > q_2$  and the rewrite system contains a rule that has  $f(q_1)(X_1, \dots, X_{i-1}, f(q_2)(\dots), X_{i+1}, \dots, X_k)$  on the left or right hand side.  $L_F$  is a regular tree language, and can be accepted by a regular tree grammar  $G_F$  with  $O(n)$  nonterminals and  $O(n^2)$  rules, where  $n = |\Sigma'|$ . A filter grammar for Fig. 1 looks as follows:

$$\begin{array}{l} S \rightarrow 1(S, S) \mid 2(S, Q_1) \mid 3(S, S) \mid 4 \mid \dots \mid 7 \\ Q_1 \rightarrow 2(S, Q_1) \mid 3(S, S) \mid 4 \mid \dots \mid 7 \end{array}$$

This grammar accepts all trees over  $\Sigma$  except ones in which a node with label 2 is the parent of a node with label 1, because such trees correspond to configurations in which a node with label  $a_z$  is the parent of a node with label  $a_x$ ,  $a_z$  and  $a_x$  are permutable, and  $2 > 1$ . In particular, it will accept the configurations (b), (c), and (e) in Fig. 1, but not (a) or (d).

Since regular tree languages are closed under intersection, we can compute a grammar  $G'$  such that  $L(G') = L(G) \cap L_F$ . This grammar has  $O(nk)$  nonterminals and  $O(n^2k)$  productions, where  $k$  is the number of production rules in  $G$ , and can be computed in time  $O(n^2k)$ . The relabelled grammar  $f(G')$  accepts all trees in which adjacent occurrences of permutable quantifiers are in a canonical order (sorted from lowest to highest node name). For example, the grammar  $G'$  for the example looks as follows; note that the nonterminal alphabet of  $G'$  is the product of the nonterminal alphabets of  $G$  and  $G_F$ .

$$\begin{array}{ll} \{1, 2, 3, 4, 5, 6, 7\}_S & \rightarrow 1(\{2, 4, 5\}_S, \{3, 6, 7\}_S) \\ \{1, 2, 3, 4, 5, 6, 7\}_S & \rightarrow 2(\{4\}_S, \{1, 3, 5, 6, 7\}_{Q_1}) \\ \{1, 2, 3, 4, 5, 6, 7\}_S & \rightarrow 3(\{6\}_S, \{1, 2, 4, 5, 7\}_S) \\ \{1, 3, 5, 6, 7\}_{Q_1} & \rightarrow 3(\{6\}_S, \{1, 5, 7\}_S) \\ \{1, 2, 4, 5, 7\}_S & \rightarrow 1(\{2, 4, 5\}_S, \{7\}_S) \\ \{1, 2, 4, 5, 7\}_S & \rightarrow 2(\{4\}_S, \{1, 5, 7\}_{Q_1}) \\ \{2, 4, 5\}_S & \rightarrow 2(\{4\}_S, \{5\}_{Q_1}) & \{4\}_S & \rightarrow 4 \\ \{3, 6, 7\}_S & \rightarrow 3(\{6\}_S, \{7\}_S) & \{5\}_S & \rightarrow 5 \\ \{1, 5, 7\}_S & \rightarrow 1(\{5\}_S, \{7\}_S) & \{5\}_{Q_1} & \rightarrow 5 \\ \{6\}_S & \rightarrow 6 & \{7\}_S & \rightarrow 7 \end{array}$$

Significantly, the grammar contains no productions for  $\{1, 3, 5, 6, 7\}_{Q_1}$  with terminal symbol 1, and no production for  $\{1, 5, 7\}_{Q_1}$ . This reduces the tree language accepted by  $f(G')$  to just the configurations (b), (c), and (e) in Fig. 1, i.e. exactly one

representative of every equivalence class. Notice that there are two different nonterminals,  $\{5\}_{Q_1}$  and  $\{5\}_S$ , corresponding to the subgraph  $\{5\}$ , so the intersected RTG is not a dominance chart any more. As we will see below, this increased expressivity increases the power of the redundancy elimination algorithm.

### 4.3 Evaluation

The algorithm presented here is not only more transparent than KT06, but also more powerful; for example, it will reduce the graph in Fig. 4 of Koller and Thater (2006) completely, whereas KT06 won't.

To measure the extent to which the new algorithm improves upon KT06, we compare both algorithms on the USRs in the Rondane treebank (version of January 2006). The Rondane treebank is a ‘‘Redwoods style’’ treebank (Oepen et al., 2002) containing MRS-based underspecified representations for sentences from the tourism domain, and is distributed together with the English Resource Grammar (ERG) (Copestake and Flickinger, 2000).

The treebank contains 999 MRS-nets, which we translate automatically into dominance graphs and further into RTGs; the median number of scope readings per sentence is 56. For our experiment, we consider all 950 MRS-nets with less than 650 000 configurations. We use a slightly weaker version of the rewrite system that Koller and Thater (2006) used in their evaluation.

It turns out that the median number of equivalence classes, computed by pairwise comparison of all configurations, is 8. The median number of configurations that remain after running our algorithm is also 8. By contrast, the median number after running KT06 is 11. For a more fine-grained comparison, Fig. 3 shows the percentage of USRs for which the two algorithms achieve complete reduction, i.e. retain only one reading per equivalence class. In the diagram, we have grouped USRs according to the natural logarithm of their numbers of configurations, and report the percentage of USRs in this group on which the algorithms were complete. The new algorithm dramatically outperforms KT06: In total, it reduces 96% of all USRs completely, whereas KT06 was complete only for 40%. This increase in completeness is partially due to the new algorithm's ability to use non-chart RTGs: For 28% of the sentences,

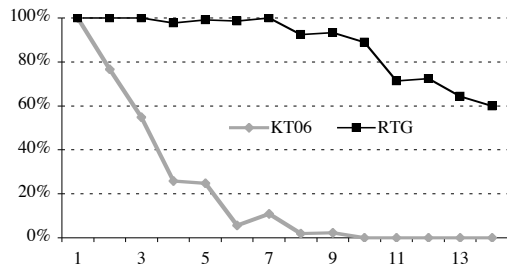


Figure 3: Percentage of USRs in Rondane for which the algorithms achieve complete reduction.

it computes RTGs that are not dominance charts. KT06 was only able to reduce 5 of these 263 graphs completely.

The algorithm needs 25 seconds to run for the entire corpus (old algorithm: 17 seconds), and it would take 50 (38) more seconds to run on the 49 large USRs that we exclude from the experiment. By contrast, it takes about 7 hours to compute the equivalence classes by pairwise comparison, and it would take an estimated several billion years to compute the equivalence classes of the excluded USRs. In short, the redundancy elimination algorithm presented here achieves nearly complete reduction at a tiny fraction of the runtime, and makes a useful task that was completely infeasible before possible.

#### 4.4 Compactness

Finally, let us briefly consider the ramifications of expressive completeness on efficiency. Ebert (2005) proves that no expressively complete underspecification formalism can be *compact*, i.e. in the worst case, the USR of a set of readings become exponentially large in the number of scope-bearing operators. In the case of RTGs, this worst case is achieved by grammars of the form  $S \rightarrow t_1 \mid \dots \mid t_n$ , where  $t_1, \dots, t_n$  are the trees we want to describe. This grammar is as big as the number of readings, i.e. worst-case exponential in the number  $n$  of scope-bearing operators, and essentially amounts to a meta-level disjunction over the readings.

Ebert takes the incompatibility between compactness and expressive completeness as a fundamental problem for underspecification. We don't see things quite as bleakly. Expressions of natural language itself are (extremely underspecified) descriptions of sets of semantic representations, and so Ebert's argument applies to NL expressions as well. This

means that describing a given set of readings may require an exponentially long discourse. Ebert's definition of compactness may be too harsh: An USR, although exponential-size in the number of quantifiers, may still be polynomial-size in the length of the discourse in the worst case.

Nevertheless, the tradeoff between compactness and expressive power is important for the design of underspecification formalisms, and RTGs offer a unique answer. They are expressively complete; but as we have seen in Fig. 2, the RTGs that are derived by semantic construction are compact, and even intersecting them with filter grammars for redundancy elimination only blows up their sizes by a factor of  $O(n^2)$ . As we add more and more information to an RTG to reduce the set of readings, ultimately to those readings that were meant in the actual context of the utterance, the grammar will become less and less compact; but this trend is counterbalanced by the overall reduction in the number of readings. For the USRs in Rondane, the intersected RTGs are, on average, 6% smaller than the original charts. Only 30% are larger than the charts, by a maximal factor of 3.66. Therefore we believe that the theoretical non-compactness should not be a major problem in a well-designed practical system.

## 5 Computing best configurations

A second advantage of using RTGs as an underspecification formalism is that we can apply existing algorithms for computing the best derivations of *weighted* regular tree grammars to compute best (that is, cheapest or most probable) configurations. This gives us the first efficient algorithm for computing the preferred reading of a scope ambiguity.

We define weighted dominance graphs and weighted tree grammars, show how to translate the former into the latter and discuss an example.

### 5.1 Weighted dominance graphs

A *weighted* dominance graph  $D = (V, E_T \uplus E_D \uplus W_D \uplus W_I)$  is a dominance graph with two new types of edges – *soft dominance edges*,  $W_D$ , and *soft disjointness edges*,  $W_I$  –, each of which is equipped with a numeric weight. Soft dominance and disjointness edges provide a mechanism for assigning weights to configurations; a soft dominance edge ex-

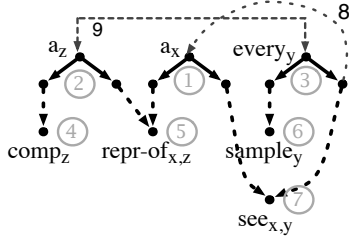


Figure 4: The graph of Fig. 1 with soft constraints

presses a preference that two nodes dominate each other in a configuration, whereas a soft disjointness edge expresses a preference that two nodes are *disjoint*, i.e. neither dominates the other.

We take the *hard backbone* of  $D$  to be the ordinary dominance graph  $B(D) = (V, E_T \uplus E_D)$  obtained by removing all soft edges. The set of *configurations* of a weighted graph  $D$  is the set of configurations of its hard backbone. For each configuration  $t$  of  $D$ , we define the *weight*  $c(t)$  to be the product of the weights of all soft dominance and disjointness edges that are satisfied in  $t$ . We can then ask for configurations of maximal weight.

Weighted dominance graphs can be used to encode the standard models of scope preferences (Pafel, 1997; Higgins and Sadock, 2003). For example, Higgins and Sadock (2003) present a machine learning approach for determining pairwise preferences as to whether a quantifier  $Q_1$  dominates another quantifier  $Q_2$ ,  $Q_2$  dominates  $Q_1$ , or neither (i.e. they are disjoint). We can represent these numbers as the weights of soft dominance and disjointness edges. An example (with artificial weights) is shown in Fig. 4; we draw the soft dominance edges as curved dotted arrows and the soft disjointness edges as angled double-headed arrows. Each soft edge is annotated with its weight. The hard backbone of this dominance graph is our example graph from Fig. 1, so it has the same five configurations. The weighted graph assigns a weight of 8 to configuration (a), a weight of 1 to (d), and a weight of 9 to (e); this is also the configuration of maximum weight.

## 5.2 Weighted tree grammars

In order to compute the maximal-weight configuration of a weighted dominance graph, we will first translate it into a *weighted regular tree grammar*. A weighted regular tree grammar (wRTG) (Graehl and Knight, 2004) is a 5-tuple  $G = (S, N, \Sigma, R, c)$  such

that  $G' = (S, N, \Sigma, R)$  is a regular tree grammar and  $c : R \rightarrow \mathbb{R}$  is a function that assigns each production rule a weight.  $G$  accepts the same language of trees as  $G'$ . It assigns each derivation a cost equal to the product of the costs of the production rules used in this derivation, and it assigns each tree in the language a cost equal to the sum of the costs of its derivations. Thus wRTGs define weights in a way that is extremely similar to PCFGs, except that we don't require any weights to sum to one.

Given a weighted, hypernormally connected dominance graph  $D$ , we can extend the chart of  $B(D)$  to a wRTG by assigning rule weights as follows: The weight of a rule  $D_0 \rightarrow i(D_1, \dots, D_n)$  is the product over the weights of all soft dominance and disjointness edges that are established by this rule. We say that a rule establishes a soft dominance edge from  $u$  to  $v$  if  $u = i$  and  $v$  is in one of the subgraphs  $D_1, \dots, D_n$ ; we say that it establishes a soft disjointness edge between  $u$  and  $v$  if  $u$  and  $v$  are in different subgraphs  $D_j$  and  $D_k$  ( $j \neq k$ ). It can be shown that the weight this grammar assigns to each derivation is equal to the weight that the original dominance graph assigns to the corresponding configuration.

If we apply this construction to the example graph in Fig. 4, we obtain the following wRTG:

$$\begin{array}{ll}
 \{1, \dots, 7\} \rightarrow a_x(\{2, 4, 5\}, \{3, 6, 7\}) & [9] \\
 \{1, \dots, 7\} \rightarrow a_z(\{4\}, \{1, 3, 5, 6, 7\}) & [1] \\
 \{1, \dots, 7\} \rightarrow every_y(\{6\}, \{1, 2, 4, 5, 7\}) & [8] \\
 \{2, 4, 5\} \rightarrow a_z(\{4\}, \{5\}) & [1] \\
 \{3, 6, 7\} \rightarrow every_y(\{6\}, \{7\}) & [1] \\
 \{1, 3, 5, 6, 7\} \rightarrow a_x(\{5\}, \{3, 6, 7\}) & [1] \\
 \{1, 3, 5, 6, 7\} \rightarrow every_y(\{6\}, \{1, 5, 7\}) & [8] \\
 \{1, 2, 4, 5, 7\} \rightarrow a_x(\{2, 4, 5\}, \{7\}) & [1] \\
 \{1, 2, 4, 5, 7\} \rightarrow a_z(\{4\}, \{1, 5, 7\}) & [1] \\
 \{1, 5, 7\} \rightarrow a_x(\{5\}, \{7\}) & [1] \\
 \{4\} \rightarrow comp_z & [1] \quad \{5\} \rightarrow repr-of_{x,z} & [1] \\
 \{6\} \rightarrow sample_y & [1] \quad \{7\} \rightarrow see_{x,y} & [1]
 \end{array}$$

For example, picking “ $a_z$ ” as the root of a configuration (Fig. 1 (c), (d)) of the entire graph has a weight of 1, because this rule establishes no soft edges. On the other hand, choosing “ $a_x$ ” as the root has a weight of 9, because this establishes the soft disjointness edge (and in fact, leads to the derivation of the maximum-weight configuration in Fig. 1 (e)).

## 5.3 Computing the best configuration

The problem of computing the best configuration of a weighted dominance graph – or equivalently, the

best derivation of a weighted tree grammar – can now be solved by standard algorithms for wRTGs. For example, Knight and Graehl (2005) present an algorithm to extract the best derivation of a wRTG in time  $O(t + n \log n)$  where  $n$  is the number of nonterminals and  $t$  is the number of rules. In practice, we can extract the best reading of the most ambiguous sentence in the Rondane treebank ( $4.5 \times 10^{12}$  readings, 75 000 grammar rules) with random soft edges in about a second.

However, notice that this is not the same problem as computing the best *tree* in the language accepted by a wRTG, as trees may have multiple derivations. The problem of computing the best tree is NP-complete (Sima'an, 1996). However, if the weighted regular tree automaton corresponding to the wRTG is deterministic, every tree has only one derivation, and thus computing best trees becomes easy again. The tree automata for dominance charts are always deterministic, and the automata for RTGs as in Section 3.2 (whose terminals correspond to the graph's node labels) are also typically deterministic if the variable names are part of the quantifier node labels. Furthermore, there are algorithms for determinizing weighted tree automata (Borchardt and Vogler, 2003; May and Knight, 2006), which could be applied as preprocessing steps for wRTGs.

## 6 Conclusion

In this paper, we have shown how regular tree grammars can be used as a formalism for scope underspecification, and have exploited the power of this view in a novel, simpler, and more complete algorithm for redundancy elimination and the first efficient algorithm for computing the best reading of a scope ambiguity. In both cases, we have adapted standard algorithms for RTGs, which illustrates the usefulness of using such a well-understood formalism. In the worst case, the RTG for a scope ambiguity is exponential in the number of scope bearers in the sentence; this is a necessary consequence of their expressive completeness. However, those RTGs that are computed by semantic construction and redundancy elimination remain compact.

Rather than showing how to do semantic construction for RTGs, we have presented an algorithm that computes RTGs from more standard underspecifica-

tion formalisms. We see RTGs as an “underspecification assembly language” – they support efficient and useful algorithms, but direct semantic construction may be inconvenient, and RTGs will rather be obtained by “compiling” higher-level underspecified representations such as dominance graphs or MRS.

This perspective also allows us to establish a connection to approaches to semantic construction which use chart-based packing methods rather than dominance-based underspecification to manage scope ambiguities. For instance, both Combinatory Categorical Grammars (Steedman, 2000) and synchronous grammars (Nesson and Shieber, 2006) represent syntactic and semantic ambiguity as part of the same parse chart. These parse charts can be seen as regular tree grammars that accept the language of parse trees, and conceivably an RTG that describes only the semantic and not the syntactic ambiguity could be automatically extracted. We could thus reconcile these completely separate approaches to semantic construction within the same formal framework, and RTG-based algorithms (e.g., for redundancy elimination) would apply equally to dominance-based and chart-based approaches. Indeed, for one particular grammar formalism it has even been shown that the parse chart contains an isomorphic image of a dominance chart (Koller and Rambow, 2007).

Finally, we have only scratched the surface of what can be done with the computation of best configurations in Section 5. The algorithms generalize easily to weights that are taken from an arbitrary ordered semiring (Golan, 1999; Borchardt and Vogler, 2003) and to computing minimal-weight rather than maximal-weight configurations. It is also useful in applications beyond semantic construction, e.g. in discourse parsing (Regneri et al., 2008).

**Acknowledgments.** We have benefited greatly from fruitful discussions on weighted tree grammars with Kevin Knight and Jonathan Graehl, and on discourse underspecification with Markus Egg. We also thank Christian Ebert, Marco Kuhlmann, Alex Lascarides, and the reviewers for their comments on the paper. Finally, we are deeply grateful to our former colleague Joachim Niehren, who was a great fan of tree automata before we even knew what they are.

## References

- E. Althaus, D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. 2003. An efficient graph algorithm for dominance constraints. *J. Algorithms*, 48:194–219.
- B. Borchardt and H. Vogler. 2003. Determinization of finite state weighted tree automata. *Journal of Automata, Languages and Combinatorics*, 8(3):417–463.
- J. Bos. 1996. Predicate logic unplugged. In *Proceedings of the Tenth Amsterdam Colloquium*, pages 133–143.
- R. P. Chaves. 2003. Non-redundant scope disambiguation in underspecified semantics. In *Proceedings of the 8th ESSLLI Student Session*, pages 47–58, Vienna.
- H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2007. Tree automata techniques and applications. Available on: <http://www.grappa.univ-lille3.fr/tata>.
- A. Copestake and D. Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Conference on Language Resources and Evaluation*.
- A. Copestake, D. Flickinger, C. Pollard, and I. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3:281–332.
- C. Ebert. 2005. *Formal investigations of underspecified representations*. Ph.D. thesis, King’s College, London.
- M. Egg, A. Koller, and J. Niehren. 2001. The Constraint Language for Lambda Structures. *Logic, Language, and Information*, 10:457–485.
- D. Flickinger, A. Koller, and S. Thater. 2005. A new well-formedness criterion for semantics debugging. In *Proceedings of the 12th HPSG Conference*, Lisbon.
- J. S. Golan. 1999. *Semirings and their applications*. Kluwer, Dordrecht.
- J. Graehl and K. Knight. 2004. Training tree transducers. In *HLT-NAACL 2004*, Boston.
- D. Higgins and J. Sadock. 2003. A machine learning approach to modeling scope preferences. *Computational Linguistics*, 29(1).
- K. Knight and J. Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Computational linguistics and intelligent text processing*, pages 1–24. Springer.
- A. Koller and J. Niehren. 2000. On underspecified processing of dynamic semantics. In *Proceedings of COLING-2000*, Saarbrücken.
- A. Koller and O. Rambow. 2007. Relating dominance formalisms. In *Proceedings of the 12th Conference on Formal Grammar*, Dublin.
- A. Koller and S. Thater. 2005a. Efficient solving and exploration of scope ambiguities. Proceedings of the ACL-05 Demo Session.
- A. Koller and S. Thater. 2005b. The evolution of dominance constraint solvers. In *Proceedings of the ACL-05 Workshop on Software*.
- A. Koller and S. Thater. 2006. An improved redundancy elimination algorithm for underspecified descriptions. In *Proceedings of COLING/ACL-2006*, Sydney.
- J. May and K. Knight. 2006. A better n-best list: Practical determinization of weighted finite tree automata. In *Proceedings of HLT-NAACL*.
- R. Nesson and S. Shieber. 2006. Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*.
- J. Niehren and S. Thater. 2003. Bridging the gap between underspecification formalisms: Minimal recursion semantics as dominance constraints. In *Proceedings of ACL 2003*.
- S. Oepen, K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants. 2002. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING’02)*, pages 1253–1257.
- J. Pafel. 1997. Skopus und logische Struktur: Studien zum Quantorenskopos im Deutschen. Habilitationsschrift, Eberhard-Karls-Universität Tübingen.
- M. Regneri, M. Egg, and A. Koller. 2008. Efficient processing of underspecified discourse representations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT) – Short Papers*, Columbus, Ohio.
- U. Reyle. 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 10(1).
- S. Shieber. 2006. Unifying synchronous tree-adjointing grammars and tree transducers via bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, Trento, Italy.
- K. Sima’an. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *Proceedings of the 16th conference on Computational linguistics*, pages 1175–1180, Morristown, NJ, USA. Association for Computational Linguistics.
- M. Steedman. 2000. *The syntactic process*. MIT Press.
- E. Vestre. 1991. An algorithm for generating non-redundant quantifier scopings. In *Proc. of EACL*, pages 251–256, Berlin.

# Classification of Semantic Relationships between Nominals Using Pattern Clusters

**Dmitry Davidov**

ICNC

Hebrew University of Jerusalem

dmitry@alice.nc.huji.ac.il

**Ari Rappoport**

Institute of Computer Science

Hebrew University of Jerusalem

arir@cs.huji.ac.il

## Abstract

There are many possible different semantic relationships between nominals. Classification of such relationships is an important and difficult task (for example, the well known noun compound classification task is a special case of this problem). We propose a novel *pattern clusters* method for nominal relationship (NR) classification. Pattern clusters are discovered in a large corpus independently of any particular training set, in an unsupervised manner. Each of the extracted clusters corresponds to some unspecified semantic relationship. The pattern clusters are then used to construct features for training and classification of specific inter-nominal relationships. Our NR classification evaluation strictly follows the ACL SemEval-07 Task 4 datasets and protocol, obtaining an f-score of 70.6, as opposed to 64.8 of the best previous work that did not use the manually provided WordNet sense disambiguation tags.

## 1 Introduction

Automatic extraction and classification of semantic relationships is a major field of activity, of both practical and theoretical interest. A prominent type of semantic relationships is that holding between nominals<sup>1</sup>. For example, in noun compounds many different semantic relationships are encoded by the same simple form (Girju et al., 2005): ‘dog food’ denotes food consumed by dogs, while ‘summer morn-

<sup>1</sup>Our use of the term ‘nominal’ follows (Girju et al., 2007), and includes simple nouns, noun compounds and multiword expressions serving as nouns.

ing’ denotes a morning that happens in the summer. These two relationships are completely different semantically but are similar syntactically, and distinguishing between them could be essential for NLP applications such as question answering and machine translation.

Relation classification usually relies on a training set in the form of tagged data. To improve results, some systems utilize additional manually constructed semantic resources such as WordNet (WN) (Beamer et al., 2007). However, in many domains and languages such resources are not available. Furthermore, usage of such resources frequently requires disambiguation and connection of the data to the resource (word sense disambiguation in the case of WordNet). Manual disambiguation is unfeasible in many practical tasks, and an automatic one may introduce errors and greatly degrade performance. It thus makes sense to try to minimize the usage of such resources, and utilize only corpus contexts in which the relevant words appear.

A leading method for utilizing context information for classification and extraction of relationships is that of patterns (Hearst, 1992; Pantel and Pannacchiotti, 2006). The standard classification process is to find in an auxiliary corpus a set of patterns in which a given training word pair co-appears, and use pattern-word pair co-appearance statistics as features for machine learning algorithms.

In this paper we introduce a novel approach, based on utilizing pattern clusters that are prepared separately and independently of the training set. We do not utilize any manually constructed resource or any manual tagging of training data beyond the cor-

rect classification, thus making our method applicable to fully automated tasks and less domain and language dependent. Moreover, our pattern clustering algorithm is fully unsupervised.

Our method is based on the observation that while each lexical pattern can be highly ambiguous, several patterns in conjunction can reliably define and represent a lexical relationship. Accordingly, we construct pattern clusters from a large generic corpus, each such cluster potentially representing some important generic relationship. This step is done without accessing any training data, anticipating that most meaningful relationships, including those in a given classification problem, will be represented by some of the discovered clusters. We then use the training set to label some of the clusters, and the labeled clusters to assign classes to tested items. One of the advantages of our method is that it can be used not only for classification, but also for further analysis and retrieval of the observed relationships<sup>2</sup>.

The semantic relationships between the components of noun compounds and between nominals in general are not easy to categorize rigorously. Several different relationship hierarchies have been proposed (Nastase and Szpakowicz, 2003; Moldovan et al., 2004). Some classes, like Container-Contained, Time-Event and Product-Producer, appear in several classification schemes, while classes like Tool-Object are more vaguely defined and are subdivided differently. Recently, SemEval-07 Task 4 (Girju et al., 2007) proposed a benchmark dataset that includes a subset of 7 widely accepted nominal relationship (NR) classes, allowing consistent evaluation of different NR classification algorithms. In the SemEval event, 14 research teams evaluated their algorithms using this benchmark. Some of the teams have used the manually annotated WN labels provided with the dataset, and some have not.

We evaluated our algorithm on SemEval-07 Task 4 data, showing superior results over participating algorithms that did not utilize WordNet disambiguation tags. We also show how pattern clusters can be used for a completely unsupervised classification of

---

<sup>2</sup>In (Davidov and Rappoport, 2008) we focus on the pattern cluster resource type itself, presenting an evaluation of its intrinsic quality based on SAT tests. In the present paper we focus on showing how the resource can be used to improve a known NLP task.

the test set. Since in this case no training data is used, this allows the automated discovery of a potentially unbiased classification scheme.

Section 2 discusses related work, Section 3 outlines the pattern clustering algorithm, Section 4 details three classification methods, and Sections 5 and 6 describe the evaluation protocol and results.

## 2 Related Work

Numerous methods have been devised for classification of semantic relationships, among which those holding between nominals constitute a prominent category. Major differences between these methods include available resources, degree of preprocessing, features used, classification algorithm and the nature of training/test data.

### 2.1 Available Resources

Many relation classification algorithms utilize WordNet. Among the 15 systems presented by the 14 SemEval teams, some utilized the manually provided WordNet tags for the dataset pairs (e.g., (Beamer et al., 2007)). In all cases, usage of WN tags improves the results significantly. Some other systems that avoided using the labels used WN as a supporting resource for their algorithms (Costello, 2007; Nakov and Hearst, 2007; Kim and Baldwin, 2007). Only three avoided WN altogether (Hendrickx et al., 2007; Bedmar et al., 2007; Aramaki et al., 2006).

Other resources used for relationship discovery include Wikipedia (Strube and Ponzetto, 2006), thesauri or synonym sets (Turney, 2005) and domain-specific semantic hierarchies like MeSH (Rosario and Hearst, 2001).

While usage of these resources is beneficial in many cases, high quality word sense annotation is not easily available. Besides, lexical resources are not available for many languages, and their coverage is limited even for English when applied to some restricted domains. In this paper we do not use any manually annotated resources apart from the classification training set.

### 2.2 Degree of Preprocessing

Many relationship classification methods utilize some language-dependent preprocessing, like deep or shallow parsing, part of speech tagging and



named entity annotation (Pantel et al., 2004). While the obtained features were shown to improve classification performance, they tend to be language dependent and error-prone when working on unusual text domains and are also highly computationally intensive when processing large corpora. To make our approach as language independent and efficient as possible, we avoided using any such preprocessing techniques.

### 2.3 Classification Features

A wide variety of features are used by different algorithms, ranging from simple bag-of-words frequencies to WordNet-based features (Moldovan et al., 2004). Several studies utilize syntactic features. Many other works manually develop a set of heuristic features devised with some specific relationship in mind, like a WordNet-based meronymy feature (Bedmar et al., 2007) or size-of feature (Aramaki et al., 2006). However, the most prominent feature type is based on lexico-syntactic patterns in which the related words co-appear.

Since (Hearst, 1992), numerous works have used patterns for discovery and identification of instances of semantic relationships (e.g., (Girju et al., 2006; Snow et al., 2006; Banko et al., 2007)). Rosenfeld and Feldman (2007) discover relationship instances by clustering entities appearing in similar contexts. Strategies were developed for discovery of multiple patterns for some specified lexical relationship (Pantel and Pennacchiotti, 2006) and for unsupervised pattern ranking (Turney, 2006). Davidov et al. (2007) use pattern clusters to define general relationships, but these are specific to a given concept. No study so far has proposed a method to define, discover and represent general relationships present in an arbitrary corpus.

In (Davidov and Rappoport, 2008) we present an approach to extract pattern clusters from an untagged corpus. Each such cluster represents some unspecified lexical relationship. In this paper, we use these pattern clusters as the (only) source of machine learning features for a nominal relationship classification problem. Unlike the majority of current studies, we avoid using any other features that require some language-specific information or are devised for specific relationship types.

### 2.4 Classification Algorithm

Various learning algorithms have been used for relation classification. Common choices include variations of SVM (Girju et al., 2004; Nastase et al., 2006), decision trees and memory-based learners. Freely available tools like Weka (Witten and Frank, 1999) allow easy experimentation with common learning algorithms (Hendrickx et al., 2007). In this paper we did not focus on a single ML algorithm, letting algorithm selection be automatically based on cross-validation results on the training set, as in (Hendrickx et al., 2007) but using more algorithms and allowing a more flexible parameter choice.

### 2.5 Training Data

As stated above, several categorization schemes for nominals have been proposed. Nastase and Szpakowicz (2003) proposed a two-level hierarchy with 5 (30) classes at the top (bottom) levels<sup>3</sup>. This hierarchy and a corresponding dataset were used in (Turney, 2005; Turney, 2006) and (Nastase et al., 2006) for evaluation of their algorithms. Moldovan et al. (2004) proposed a different scheme with 35 classes. The most recent dataset has been developed for SemEval 07 Task 4 (Girju et al., 2007). This manually annotated dataset includes a representative rather than exhaustive list of 7 important nominal relationships. We have used this dataset, strictly following the evaluation protocol. This made it possible to meaningfully compare our method to state-of-the-art methods for relation classification.

## 3 Pattern Clustering Algorithm

Our pattern clustering algorithm is designed for the unsupervised definition and discovery of generic semantic relationships. The algorithm first discovers and clusters patterns in which a single ('hook') word participates, and then merges the resulting clusters to form the final structure. In (Davidov and Rappoport, 2008) we describe the algorithm at length, discuss its behavior and parameters in detail, and evaluate its intrinsic quality. To assist readers of the present paper, in this section we provide an overview. Examples of some resulting pattern clusters are given in Section 6. We refer to a pattern

<sup>3</sup>Actually, there were 50 relationships at the bottom level, but valid nominal instances were found only for 30.

contained in our clusters (a pattern type) as a ‘pattern’ and to an occurrence of a pattern in the corpus (a pattern token) as a ‘pattern instance’.

The algorithm does not rely on any data from the classification training set, hence we do not need to repeat its execution for different classification problems. To calibrate its parameters, we ran it a few times with varied parameters settings, producing several different configurations of pattern clusters with different degrees of noise, coverage and granularity. We then chose the best configuration for our task automatically without re-running pattern clustering for each specific problem (see Section 5.3).

### 3.1 Hook Words and Hook Corpora

As a first step, we randomly sample a set of hook words, which will be used in order to discover relationships that generally occur in the corpus. To avoid selection of ambiguous words or typos, we do not select words with frequency higher than a parameter  $F_C$  and lower than a threshold  $F_B$ . We also limit the total number  $N$  of hook words. For each hook word, we now create a *hook corpus*, the set of the contexts in which the word appears. Each context is a window containing  $W$  words or punctuation characters before and after the hook word.

### 3.2 Pattern Specification

To specify patterns, following (Davidov and Rapoport, 2006) we classify words into high-frequency words (HFWs) and content words (CWs). A word whose frequency is more (less) than  $F_H$  ( $F_C$ ) is considered to be a HFW (CW). Our patterns have the general form

**[Prefix]**  $CW_1$  **[Infix]**  $CW_2$  **[Postfix]**

where Prefix, Infix and Postfix contain only HFWs. We require Prefix and Postfix to be a single HFW, while Infix can contain any number of HFWs (limiting pattern length by window size). This form may include patterns like ‘*such X as Y and*’. At this stage, the pattern slots can contain only single words; however, when using the final pattern clusters for nominal relationship classification, slots can contain multiword nominals.

### 3.3 Discovery of Target Words

For each of the hook corpora, we now extract all pattern instances where one CW slot contains the

hook word and the other CW slot contains some other (‘target’) word. To avoid the selection of common words as target words, and to avoid targets appearing in pattern instances that are relatively fixed multiword expressions, we sort all target words in a given hook corpus by pointwise mutual information between hook and target, and drop patterns obtained from pattern instances containing the lowest and highest  $L$  percent of target words.

### 3.4 Pattern Clustering

We now have for each hook corpus a set of patterns, together with the target words used for their extraction, and we want to cluster pattern types. First, we group in clusters all patterns extracted using the same target word. Second, we merge clusters that share more than  $S$  percent of their patterns. Some patterns can appear in more than a single cluster. Finally, we merge pattern clusters from different hook corpora, to avoid clusters specific to a single hook word. During merging, we define and utilize *core patterns* and *unconfirmed patterns*, which are weighed differently during cluster labeling (see Section 4.2). We merge clusters from different hook corpora using the following algorithm:

1. Remove all patterns originating from a single hook corpus only.
2. Mark all patterns of all present clusters as unconfirmed.
3. While there exists some cluster  $C_1$  from corpus  $D_X$  containing only unconfirmed patterns:
  - (a) Select a cluster with a minimal number of patterns.
  - (b) For each corpus  $D$  different from  $D_X$ :
    - i. Scan  $D$  for clusters  $C_2$  that share at least  $S$  percent of their patterns, and all of their core patterns, with  $C_1$ .
    - ii. Add all patterns of  $C_2$  to  $C_1$ , setting all shared patterns as core and all others as unconfirmed.
    - iii. Remove cluster  $C_2$ .
  - (c) If all of  $C_1$ ’s patterns remain unconfirmed remove  $C_1$ .
4. If several clusters have the same set of core patterns merge them according to rules (i,ii).

At the end of this stage, we have a set of pattern clusters where for each cluster there are two subsets, core patterns and unconfirmed patterns.

## 4 Relationship Classification

Up to this stage we did not access the training set in any way and we did not use the fact that the target relations are those holding between nominals. Hence, only a small part of the acquired pattern clusters may be relevant for a given NR classification task, while other clusters can represent completely different relationships (e.g., between verbs). We now use the acquired clusters to learn a model for the given labeled training set and to use this model for classification of the test set. First we describe how we deal with data sparseness. Then we propose a HITS measure used for cluster labeling, and finally we present three different classification methods that utilize pattern clusters.

### 4.1 Enrichment of Provided Data

Our classification algorithm is based on contexts of given nominal pairs. Co-appearance of nominal pairs can be very rare (in fact, some word pairs in the Task 4 set co-appear only once in Yahoo web search). Hence we need more contexts where the given nominals or nominals similar to them co-appear. This step does not require the training labels (the correct classifications), so we do it for both training and test pairs. We do it in two stages: extracting similar nominals, and obtaining more contexts.

#### 4.1.1 Extracting more words

For each nominal pair  $(w_1, w_2)$  in a given sentence  $S$ , we use a method similar to (Davidov and Rappoport, 2006) to extract words that have a shared meaning with  $w_1$  or  $w_2$ . We discover such words by scanning our corpora and querying the web for symmetric patterns (obtained automatically from the corpus as in (Davidov and Rappoport, 2006)) that contain  $w_1$  or  $w_2$ . To avoid getting instances of  $w_{1,2}$  with a different meaning, we also require that the second word will appear in the same text paragraph or the same web page. For example, if we are given a pair  $\langle \text{loans}, \text{students} \rangle$  and we see a sentence ‘... *loans and scholarships for students and professionals ...*’, we use the symmetric pattern ‘X and Y’ to add the word *scholarships* to the group of *loans* and to add the word *professionals* to the group of *students*. We do not take words from the sentence ‘*In European soccer there are transfers and*

*loans...*’ since its context does not contain the word *students*. In cases where there are only several or zero instances where the two nominals co-appear, we dismiss the latter rule and scan for each nominal separately. Note that ‘loans’ can also be a verb, so usage of a part-of-speech tagger might reduce noise.

If the number of instances for a desired nominal is very low, our algorithm trims the first words in these nominal and repeats the search (e.g.,  $\langle \text{simulation study}, \text{voluminous results} \rangle$  becomes  $\langle \text{study}, \text{results} \rangle$ ). This step is the only one specific to English, using the nature of English noun compounds. Our desire in this case is to keep the head words.

#### 4.1.2 Extracting more contexts using the new words

To find more instances where nominals similar to  $w_1$  and  $w_2$  co-appear in HFW patterns, we construct web queries using combinations of each nominal’s group and extract patterns from the search result snapshots (the two line summary provided by search engines for each search result).

### 4.2 The HITS Measure

To use clusters for classification we define a HITS measure similar to that of (Davidov et al., 2007), reflecting the affinity of a given nominal pair to a given cluster. We use the pattern clusters from Section 3 and the additional data collected during the enrichment phase to estimate a HITS value for each cluster and each pair in the training and test sets. For a given nominal pair  $(w_1, w_2)$  and cluster  $C$  with  $n$  core patterns  $P_{core}$  and  $m$  unconfirmed patterns  $P_{unconf}$ ,

$$\begin{aligned} \text{HITS}(C, (w_1, w_2)) &= \\ &|\{p; (w_1, w_2) \text{ appears in } p \in P_{core}\}| / n + \\ &\alpha \times |\{p; (w_1, w_2) \text{ appears in } p \in P_{unconf}\}| / m. \end{aligned}$$

In this formula, ‘appears in’ means that the nominal pair appears in instances of this pattern extracted from the original corpus or retrieved from the web at the previous stage. Thus if some pair appears in most of the patterns of some cluster it receives a high HITS value for this cluster.  $\alpha$  (0..1) is a parameter that lets us modify the relative weight of core and unconfirmed patterns.

### 4.3 Classification Using Pattern Clusters

We present three ways to use pattern clusters for relationship classification.

#### 4.3.1 Classification by cluster labeling

One way to train a classifier in our case is to attach a single relationship label to each cluster during the training phase, and to assign each unlabeled pair to some labeled cluster during the test phase. We use the following normalized HITS measure to label the involved pattern clusters. Denote by  $k_i$  the number of training pairs in class  $i$  in training set  $T$ . Then

$$Label(C) = \underset{p \in T, Label(p)=i}{argmax_i} \sum hits(C, p) / k_i$$

Clusters where the above sum is zero remain unlabeled. In the test phase we assign to each test pair  $p$  the label of the labeled cluster  $C$  that received the highest  $HITS(C, p)$  value. If there are several clusters with a highest HITS value, then the algorithm selects a ‘clarifying’ set of patterns – patterns that are different in these best clusters. Then it constructs clarifying web queries that contain the test nominal pair inside the clarifying patterns. The effect is to increment the HITS value of the cluster containing a clarifying pattern if an appropriate pattern instance (including the target nominals) was found on the web. We start with the most frequent clarifying pattern and perform additional queries until no clarifying patterns are left or until some labeled cluster obtains a highest HITS value. If no patterns are left but there are still several winning clusters, we assign to the pair the label of the cluster with the largest number of pattern instances in the corpus.

One advantage of this method is that we get as a by-product a set of labeled pattern clusters. Examination of this set can help to distinguish and analyze (by means of patterns) which different relationships actually exist for each class in the training set. Furthermore, labeled pattern clusters can be used for web queries to obtain additional examples of the same relationship.

#### 4.3.2 Classification by cluster HITS values as features

In this method we treat the HITS measure for a cluster as a feature for a machine learning classification

algorithm. To do this, we construct feature vectors from each training pair, where each feature is the HITS measure corresponding to a single pattern cluster. We prepare test vectors similarly. Once we have feature vectors, we can use a variety of classifiers (we used those in Weka) to construct a model and to evaluate it on the test set.

#### 4.3.3 Unsupervised clustering

If we are not given any training set, it is still possible to separate between different relationship types by grouping the feature vectors of Section 4.3.2 into clusters. This can be done by applying k-means or another clustering algorithm to the feature vectors described above. This makes the whole approach completely unsupervised. However, it does not provide any inherent labeling, making an evaluation difficult.

## 5 Experimental Setup

The main problem in a fair evaluation of NR classification is that there is no widely accepted list of possible relationships between nominals. In our evaluation we have selected the setup and data from SemEval-07 Task 4 (Girju et al., 2007). Selecting this type of dataset allowed us to compare to 6 submitted state-of-art systems that evaluated on exactly the same data and to 9 other systems that utilize additional information (WN labels). We have applied our three different classification methods on the given data set.

### 5.1 SemEval-07 Task 4 Overview

Task 4 (Girju et al., 2007) involves classification of relationships between simple nominals other than named entities. Seven distinct relationships were chosen: Cause-Effect, Instrument-Agency, Product-Producer, Origin-Entity, Theme-Tool, Part-Whole, and Content-Container. For each relationship, the provided dataset consists of 140 training and 70 test examples. Examples were binary tagged as belonging/not belonging to the tested relationship. The vast majority of negative examples were near-misses, acquired from the web using the same lexico-syntactic patterns as the positives. Examples appear as sentences with the nominal pair tagged. Nouns in this pair were manually labeled with their corresponding WordNet 3 labels and the web queries used to

obtain the sentences. The 15 submitted systems were assigned into 4 categories according to whether they use the WordNet and Query tags (some systems were assigned to more than a single category, since they reported experiments in several settings). In our evaluation we do not utilize WordNet or Query tags, hence we compare ourselves with the corresponding group (A), containing 6 systems.

## 5.2 Corpus and Web Access

Our algorithm uses two corpora. We estimate frequencies and perform primary search on a local web corpus containing about 68GB untagged plain text. This corpus was extracted from the web starting from open directory links, comprising English web pages with varied topics and styles (Gabrilovich and Markovitch, 2005). To enrich the set of given word pairs and patterns as described in Section 4.1 and to perform clarifying queries, we utilize the Yahoo API for web queries. For each query, if the desired words/patterns were found in a page link’s snapshot, we do not use the link, otherwise we download the page from the retrieved link and then extract the required data. If only several links were found for a given word pair we perform local crawling to depth 3 in an attempt to discover more instances.

## 5.3 Parameters and Learning Algorithm

Our algorithm utilizes several parameters. Instead of calibrating them manually, we only provided a desired range for each, and the final parameter values were obtained during selection of the best-performing setup using 10-fold cross-validation on the training set. For each parameter we have estimated its desired range using the (Nastase and Szpakowicz, 2003) set as a development set. Note that this set uses an entirely different relationship classification scheme. We ran the pattern clustering phase on 128 different sets of parameters, obtaining 128 different clustering schemes with varied granularity, noise and coverage.

The parameter ranges obtained are:  $F_C$  (meta-pattern content word frequency and upper bound for hook word selection): 100 – 5000 words per million (wpm);  $F_H$  (meta-pattern HFW): 10 – 100 wpm;  $F_B$  (low word count for hook word filtering): 1 – 50 wpm;  $N$  (number of hook words): 100 – 1000;  $W$  (window size): 5 or window = sentence;  $L$  (tar-

get word mutual information filter):  $1/3 - 1/5$ ;  $S$  (cluster overlap filter for cluster merging):  $2/3$ ;  $\alpha$  (core vs. unconfirmed weight for HITS estimation):  $0.1 - 0.01$ ;  $S$  (commonality for cluster merging):  $2/3$ . As designed, each parameter indeed influences a certain effect. Naturally, the parameters are not mutually independent. Selecting the best configuration in the cross-validation phase makes the algorithm flexible and less dependent on hard-coded parameter values.

Selection of learning algorithm and its algorithm-specific parameters were done as follows. For each of the 7 classification tasks (one per relationship type), for each of the 128 pattern clustering schemes, we prepared a list of most of the compatible algorithms available in Weka, and we automatically selected the model (a parameter set and an algorithm) which gave the best 10-fold cross-validation results. The winning algorithms were LWL (Atkeson et al., 1997), SMO (Platt, 1999), and K\* (Cleary and Trigg, 1995) (there were 7 tasks, and different algorithms could be selected for each task). We then used the obtained model to classify the testing set. This allowed us to avoid fixing parameters that are best for a specific dataset but not for others. Since each dataset has only 140 examples, the computation time of each learning algorithm is negligible.

## 6 Results

The pattern clustering phase results in 90 to 3000 distinct pattern clusters, depending on the parameter setup. Manual sampling of these clusters indeed reveals that many clusters contain patterns specific to some apparent lexical relationship. For example, we have discovered such clusters as: {‘buy  $Y$  accessory for  $X$ ’, ‘shipping  $Y$  for  $X$ ’, ‘ $Y$  is available for  $X$ ’, ‘ $Y$  are available for  $X$ ’, ‘ $Y$  are available for  $X$  systems’, ‘ $Y$  for  $X$ ’} and {‘best  $X$  for  $Y$ ’, ‘ $X$  types for  $Y$ ’, ‘ $Y$  with  $X$ ’, ‘ $X$  is required for  $Y$ ’, ‘ $X$  as required for  $Y$ ’, ‘ $X$  for  $Y$ ’}. Note that some patterns (‘ $Y$  for  $X$ ’) can appear in many clusters.

We applied the three classification methods described in Section 4.3 to Task 4 data. For supervised classification we strictly followed the SemEval datasets and rules. For unsupervised classification we did not use any training data. Using the k-means algorithm, we obtained two nearly equal unlabeled

Method	P	R	F	Acc
Unsupervised clustering (4.3.3)	64.5	61.3	62.0	64.5
Cluster Labeling (4.3.1)	65.1	69.0	67.2	68.5
HITS Features (4.3.2)	<b>69.1</b>	<b>70.6</b>	<b>70.6</b>	<b>70.1</b>
Best Task 4 (no WordNet)	66.1	66.7	64.8	66.0
Best Task 4 (with WordNet)	79.7	69.8	72.4	76.3

Table 1: Our SemEval-07 Task 4 results.

Relation Type	F	Acc	C
Cause-Effect	69.7	71.4	2
Instrument-Agency	76.5	74.2	1
Product-Producer	76.4	83.8	1
Origin-Entity	65.4	62.6	4
Theme-Tool	59.4	58.7	6
Part-Whole	74.3	70.9	1
Content-Container	72.6	69.2	2

Table 2: By-relation Task 4 HITS-based results. **C** is the number of clusters with positive labels.

clusters containing test samples. For evaluation we assigned a negative/positive label to these two clusters according to the best alignment with true labels.

Table 1 shows our results, along with the best Task 4 result not using WordNet labels (Costello, 2007). For reference, the best results overall (Beamer et al., 2007) are also shown. The table shows precision (P) recall (R), F-score (F), and Accuracy (Acc) (percentage of correctly classified examples).

We can see that while our algorithm is not as good as the best method that utilizes WordNet tags, results are superior to all participants who did not use these tags. We can also see that the unsupervised method results are above the random baseline (50%). In fact, our results (f-score 62.0, accuracy 64.5) are better than the averaged results (58.0, 61.1) of the group that did not utilize WN tags.

Table 2 shows the HITS-based classification results (F-score and Accuracy) and the number of positively labeled clusters (C) for each relation. As observed by participants of Task 4, we can see that different sets vary greatly in difficulty. However, we also obtain a nice insight as to why this happens – relations like Theme-Tool seem very ambiguous and are mapped to several clusters, while relations like Product-Producer seem to be well-defined by the obtained pattern clusters.

The SemEval dataset does not explicitly mark items whose correct classification requires analysis of the context of the whole sentence in which they appear. Since our algorithm does not utilize test sen-

tence contextual information, we do not expect it to show exceptional performance on such items. This is a good topic for future research.

Since the SemEval dataset is of a very specific nature, we have also applied our classification framework to the (Nastase and Szpakowicz, 2003) dataset, which contains 600 pairs labeled with 5 main relationship types. We have used the exact evaluation procedure described in (Turney, 2006), achieving a class f-score average of 60.1, as opposed to 54.6 in (Turney, 2005) and 51.2 in (Nastase et al., 2006). This shows that our method produces superior results for rather differing datasets.

## 7 Conclusion

Relationship classification is known to improve many practical tasks, e.g., textual entailment (Tatu and Moldovan, 2005). We have presented a novel framework for relationship classification, based on pattern clusters prepared as a standalone resource independently of the training set.

Our method outperforms current state-of-the-art algorithms that do not utilize WordNet tags on Task 4 of SemEval-07. In practical situations, it would not be feasible to provide a large amount of such sense disambiguation tags manually. Our method also shows competitive performance compared to the majority of task participants that do utilize WN tags. Our method can produce labeled pattern clusters, which can be potentially useful for automatic discovery of additional instances for a given relationship. We intend to pursue this promising direction in future work.

**Acknowledgement.** We would like to thank the anonymous reviewers, whose comments have greatly improved the quality of this paper.

## References

- Aramaki, E., Imai, T., Miyo, K., and Ohe, K., 2007. UTH: semantic relation classification using physical sizes. *ACL SemEval '07 Workshop*.
- Atkeson, C., Moore, A., and Schaal, S., 1997. Locally weighted learning. *Artificial Intelligence Review*, 11(1–5): 75–113.

- Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O., 2007. Open information extraction from the Web. *IJCAI '07*.
- Beamer, B., Bhat, S., Chee, B., Fister, A., Rozovskaya A., and Girju, R., 2007. UIUC: A knowledge-rich approach to identifying semantic relations between nominals. *ACL SemEval '07 Workshop*.
- Bedmar, I. S., Samy, D., and Martinez, J. L., 2007. UC3M: Classification of semantic relations between nominals using sequential minimal optimization. *ACL SemEval '07 Workshop*.
- Cleary, J. G., Trigg, L. E., 1995. K\*: An instance-based learner using and entropic distance measure. *ICML '95*.
- Costello, F. J., 2007. UCD-FC: Deducing semantic relations using WordNet senses that occur frequently in a database of noun-noun compounds. *ACL SemEval '07 Workshop*.
- Davidov, D., Rappoport, A., 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. *COLING-ACL '06*
- Davidov D., Rappoport A. and Koppel M., 2007. Fully unsupervised discovery of concept-specific relationships by Web mining. *ACL '07*.
- Davidov, D., Rappoport, A., 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated SAT analogy questions. *ACL '08*.
- Gabrilovich, E., Markovitch, S., 2005. Feature generation for text categorization using world knowledge. *IJCAI '05*.
- Girju, R., Giuglea, A., Olteanu, M., Fortu, O., Bolohan, O., and Moldovan, D., 2004. Support vector machines applied to the classification of semantic relations in nominalized noun phrases. *HLT/NAACL '04 Workshop on Computational Lexical Semantics*.
- Girju, R., Moldovan, D., Tatu, M., and Antohe, D., 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19(4):479-496.
- Girju, R., Badulescu, A., and Moldovan, D., 2006. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1).
- Girju, R., Hearst, M., Nakov, P., Nastase, V., Szpakowicz, S., Turney, P., and Yuret, D., 2007. Task 04: Classification of semantic relations between nominal at SemEval 2007. *4th Intl. Workshop on Semantic Evaluations (SemEval '07)*, in *ACL '07*.
- Hearst, M., 1992. Automatic acquisition of hyponyms from large text corpora. *COLING '92*
- Hendrickx, I., Morante, R., Sporleder, C., and van den Bosch, A., 2007. Machine learning of semantic relations with shallow features and almost no data. *ACL SemEval '07 Workshop*.
- Kim, S.N., Baldwin, T., 2007. MELB-KB: Nominal classification as noun compound interpretation. *ACL SemEval '07 Workshop*.
- Moldovan, D., Badulescu, A., Tatu, M., Antohe, D., and Girju, R., 2004. Models for the semantic classification of noun phrases. *HLT-NAACL '04 Workshop on Computational Lexical Semantics*.
- Nakov, P., and Hearst, M., 2007. UCB: System description for SemEval Task #4. *ACL SemEval '07 Workshop*.
- Nastase, V., Szpakowicz, S., 2003. Exploring noun-modifier semantic relations. In *Fifth Intl. Workshop on Computational Semantics (IWCS-5)*.
- Nastase, V., Sayyad-Shirabad, J., Sokolova, M., and Szpakowicz, S., 2006. Learning noun-modifier semantic relations with corpus-based and WordNet-based features. In *Proceedings of the 21st National Conference on Artificial Intelligence, Boston, MA*.
- Pantel, P., Ravichandran, D., and Hovy, E., 2004. Towards terascale knowledge acquisition. *COLING '04*.
- Pantel, P., Pennacchiotti, M., 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. *COLING-ACL '06*.
- Platt, J., 1999. Fast training of support vector machines using sequential minimal optimization. In *Scholkopf, Burges, and Smola, Advances in Kernel Methods – Support Vector Learning*, pp. 185–208. MIT Press.
- Rosario, B., Hearst, M., 2001. Classifying the semantic relations in noun compounds. *EMNLP '01*.
- Rosenfeld, B., Feldman, R., 2007. Clustering for unsupervised relation identification. *CIKM '07*.
- Snow, R., Jurafsky, D., Ng, A.Y., 2006. Semantic taxonomy induction from heterogeneous evidence. *COLING-ACL '06*.
- Strube, M., Ponzetto, S., 2006. WikiRelate! computing semantic relatedness using Wikipedia. *AAAI '06*.
- Tatu, M., Moldovan, D., 2005. A semantic approach to recognizing textual entailment. *HLT/EMNLP '05*.
- Turney, P., 2005. Measuring semantic similarity by latent relational analysis. *IJCAI '05*.
- Turney, P., 2006. Expressing implicit semantic relations without supervision. *COLING-ACL '06*.
- Witten, H., Frank, E., 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman, San Francisco, CA.

# Vector-based Models of Semantic Composition

Jeff Mitchell and Mirella Lapata

School of Informatics, University of Edinburgh  
2 Buccleuch Place, Edinburgh EH8 9LW, UK  
jeff.mitchell@ed.ac.uk, mlap@inf.ed.ac.uk

## Abstract

This paper proposes a framework for representing the meaning of phrases and sentences in vector space. Central to our approach is vector composition which we operationalize in terms of additive and multiplicative functions. Under this framework, we introduce a wide range of composition models which we evaluate empirically on a sentence similarity task. Experimental results demonstrate that the multiplicative models are superior to the additive alternatives when compared against human judgments.

## 1 Introduction

Vector-based models of word meaning (Lund and Burgess, 1996; Landauer and Dumais, 1997) have become increasingly popular in natural language processing (NLP) and cognitive science. The appeal of these models lies in their ability to represent meaning simply by using distributional information under the assumption that words occurring within similar contexts are semantically similar (Harris, 1968).

A variety of NLP tasks have made good use of vector-based models. Examples include automatic thesaurus extraction (Grefenstette, 1994), word sense discrimination (Schütze, 1998) and disambiguation (McCarthy et al., 2004), collocation extraction (Schone and Jurafsky, 2001), text segmentation (Choi et al., 2001), and notably information retrieval (Salton et al., 1975). In cognitive science vector-based models have been successful in simulating semantic priming (Lund and Burgess, 1996; Landauer and Dumais, 1997) and text comprehension (Landauer and Dumais, 1997; Foltz et al.,

1998). Moreover, the vector similarities within such semantic spaces have been shown to substantially correlate with human similarity judgments (McDonald, 2000) and word association norms (Denhire and Lemaire, 2004).

Despite their widespread use, vector-based models are typically directed at representing words in isolation and methods for constructing representations for phrases or sentences have received little attention in the literature. In fact, the commonest method for combining the vectors is to average them. Vector averaging is unfortunately insensitive to word order, and more generally syntactic structure, giving the same representation to any constructions that happen to share the same vocabulary. This is illustrated in the example below taken from Landauer et al. (1997). Sentences (1-a) and (1-b) contain exactly the same set of words but their meaning is entirely different.

- (1) a. It was not the sales manager who hit the bottle that day, but the office worker with the serious drinking problem.
- b. That day the office manager, who was drinking, hit the problem sales worker with a bottle, but it was not serious.

While vector addition has been effective in some applications such as essay grading (Landauer and Dumais, 1997) and coherence assessment (Foltz et al., 1998), there is ample empirical evidence that syntactic relations across and within sentences are crucial for sentence and discourse processing (Neville et al., 1991; West and Stanovich, 1986) and modulate cognitive behavior in sentence priming (Till et al., 1988) and inference tasks (Heit and



Rubinstein, 1994).

Computational models of semantics which use symbolic logic representations (Montague, 1974) can account naturally for the meaning of phrases or sentences. Central in these models is the notion of compositionality — the meaning of complex expressions is determined by the meanings of their constituent expressions and the rules used to combine them. Here, semantic analysis is guided by syntactic structure, and therefore sentences (1-a) and (1-b) receive distinct representations. The downside of this approach is that differences in meaning are qualitative rather than quantitative, and degrees of similarity cannot be expressed easily.

In this paper we examine models of semantic composition that are empirically grounded and can represent similarity relations. We present a general framework for vector-based composition which allows us to consider different classes of models. Specifically, we present both additive and multiplicative models of vector combination and assess their performance on a sentence similarity rating experiment. Our results show that the multiplicative models are superior and correlate significantly with behavioral data.

## 2 Related Work

The problem of vector composition has received some attention in the connectionist literature, particularly in response to criticisms of the ability of connectionist representations to handle complex structures (Fodor and Pylyshyn, 1988). While neural networks can readily represent single distinct objects, in the case of multiple objects there are fundamental difficulties in keeping track of which features are bound to which objects. For the hierarchical structure of natural language this binding problem becomes particularly acute. For example, simplistic approaches to handling sentences such as *John loves Mary* and *Mary loves John* typically fail to make valid representations in one of two ways. Either there is a failure to distinguish between these two structures, because the network fails to keep track of the fact that *John* is subject in one and object in the other, or there is a failure to recognize that both structures involve the same participants, because *John* as a subject has a distinct representation from *John* as an object. In contrast, symbolic representations can naturally handle the binding of constituents to their roles, in a systematic manner that

avoids both these problems.

Smolensky (1990) proposed the use of tensor products as a means of binding one vector to another. The tensor product  $\mathbf{u} \otimes \mathbf{v}$  is a matrix whose components are all the possible products  $u_i v_j$  of the components of vectors  $\mathbf{u}$  and  $\mathbf{v}$ . A major difficulty with tensor products is their dimensionality which is higher than the dimensionality of the original vectors (precisely, the tensor product has dimensionality  $m \times n$ ). To overcome this problem, other techniques have been proposed in which the binding of two vectors results in a vector which has the same dimensionality as its components. Holographic reduced representations (Plate, 1991) are one implementation of this idea where the tensor product is projected back onto the space of its components.

The projection is defined in terms of *circular convolution* a mathematical function that compresses the tensor product of two vectors. The compression is achieved by summing along the transdiagonal elements of the tensor product. Noisy versions of the original vectors can be recovered by means of *circular correlation* which is the approximate inverse of circular convolution. The success of circular correlation crucially depends on the components of the  $n$ -dimensional vectors  $\mathbf{u}$  and  $\mathbf{v}$  being randomly distributed with mean 0 and variance  $\frac{1}{n}$ . This poses problems for modeling linguistic data which is typically represented by vectors with non-random structure.

Vector addition is by far the most common method for representing the meaning of linguistic sequences. For example, assuming that individual words are represented by vectors, we can compute the meaning of a sentence by taking their mean (Foltz et al., 1998; Landauer and Dumais, 1997). Vector addition does not increase the dimensionality of the resulting vector. However, since it is order independent, it cannot capture meaning differences that are modulated by differences in syntactic structure. Kintsch (2001) proposes a variation on the vector addition theme in an attempt to model how the meaning of a predicate (e.g., *run*) varies depending on the arguments it operates upon (e.g. *the horse ran* vs. *the color ran*). The idea is to add not only the vectors representing the predicate and its argument but also the neighbors associated with both of them. The neighbors, Kintsch argues, can ‘strengthen features of the predicate that are appropriate for the argument of the predication’.

	animal	stable	village	gallop	jokey
horse	0	6	2	10	4
run	1	8	4	4	0

Figure 1: A hypothetical semantic space for *horse* and *run*

Unfortunately, comparisons across vector composition models have been few and far between in the literature. The merits of different approaches are illustrated with a few hand picked examples and parameter values and large scale evaluations are uniformly absent (see Frank et al. (2007) for a criticism of Kintsch’s (2001) evaluation standards). Our work proposes a framework for vector composition which allows the derivation of different types of models and licenses two fundamental composition operations, multiplication and addition (and their combination). Under this framework, we introduce novel composition models which we compare empirically against previous work using a rigorous evaluation methodology.

### 3 Composition Models

We formulate semantic composition as a function of two vectors,  $\mathbf{u}$  and  $\mathbf{v}$ . We assume that individual words are represented by vectors acquired from a corpus following any of the parametrisations that have been suggested in the literature.<sup>1</sup> We briefly note here that a word’s vector typically represents its co-occurrence with neighboring words. The construction of the semantic space depends on the definition of linguistic context (e.g., neighbouring words can be documents or collocations), the number of components used (e.g., the  $k$  most frequent words in a corpus), and their values (e.g., as raw co-occurrence frequencies or ratios of probabilities). A hypothetical semantic space is illustrated in Figure 1. Here, the space has only five dimensions, and the matrix cells denote the co-occurrence of the target words (*horse* and *run*) with the context words *animal*, *stable*, and so on.

Let  $\mathbf{p}$  denote the composition of two vectors  $\mathbf{u}$  and  $\mathbf{v}$ , representing a pair of constituents which stand in some syntactic relation  $R$ . Let  $K$  stand for any additional knowledge or information which is needed to construct the semantics of their composi-

<sup>1</sup>A detailed treatment of existing semantic space models is outside the scope of the present paper. We refer the interested reader to Padó and Lapata (2007) for a comprehensive overview.

tion. We define a general class of models for this process of composition as:

$$\mathbf{p} = f(\mathbf{u}, \mathbf{v}, R, K) \quad (1)$$

The expression above allows us to derive models for which  $\mathbf{p}$  is constructed in a distinct space from  $\mathbf{u}$  and  $\mathbf{v}$ , as is the case for tensor products. It also allows us to derive models in which composition makes use of background knowledge  $K$  and models in which composition has a dependence, via the argument  $R$ , on syntax.

To derive specific models from this general framework requires the identification of appropriate constraints to narrow the space of functions being considered. One particularly useful constraint is to hold  $R$  fixed by focusing on a single well defined linguistic structure, for example the verb-subject relation. Another simplification concerns  $K$  which can be ignored so as to explore what can be achieved in the absence of additional knowledge. This reduces the class of models to:

$$\mathbf{p} = f(\mathbf{u}, \mathbf{v}) \quad (2)$$

However, this still leaves the particular form of the function  $f$  unspecified. Now, if we assume that  $\mathbf{p}$  lies in the same space as  $\mathbf{u}$  and  $\mathbf{v}$ , avoiding the issues of dimensionality associated with tensor products, and that  $f$  is a linear function, for simplicity, of the cartesian product of  $\mathbf{u}$  and  $\mathbf{v}$ , then we generate a class of *additive* models:

$$\mathbf{p} = \mathbf{A}\mathbf{u} + \mathbf{B}\mathbf{v} \quad (3)$$

where  $\mathbf{A}$  and  $\mathbf{B}$  are matrices which determine the contributions made by  $\mathbf{u}$  and  $\mathbf{v}$  to the product  $\mathbf{p}$ . In contrast, if we assume that  $f$  is a linear function of the tensor product of  $\mathbf{u}$  and  $\mathbf{v}$ , then we obtain *multiplicative* models:

$$\mathbf{p} = \mathbf{C}\mathbf{u}\mathbf{v} \quad (4)$$

where  $\mathbf{C}$  is a tensor of rank 3, which projects the tensor product of  $\mathbf{u}$  and  $\mathbf{v}$  onto the space of  $\mathbf{p}$ .

Further constraints can be introduced to reduce the free parameters in these models. So, if we assume that only the  $i$ th components of  $\mathbf{u}$  and  $\mathbf{v}$  contribute to the  $i$ th component of  $\mathbf{p}$ , that these components are not dependent on  $i$ , and that the function is symmetric with regard to the interchange of  $\mathbf{u}$

and  $\mathbf{v}$ , we obtain a simpler instantiation of an additive model:

$$p_i = u_i + v_i \quad (5)$$

Analogously, under the same assumptions, we obtain the following simpler multiplicative model:

$$p_i = u_i \cdot v_i \quad (6)$$

For example, according to (5), the addition of the two vectors representing *horse* and *run* in Figure 1 would yield  $\mathbf{horse} + \mathbf{run} = [1 \ 14 \ 6 \ 14 \ 4]$ . Whereas their product, as given by (6), is  $\mathbf{horse} \cdot \mathbf{run} = [0 \ 48 \ 8 \ 40 \ 0]$ .

Although the composition model in (5) is commonly used in the literature, from a linguistic perspective, the model in (6) is more appealing. Simply adding the vectors  $\mathbf{u}$  and  $\mathbf{v}$  lumps their contents together rather than allowing the content of one vector to pick out the relevant content of the other. Instead, it could be argued that the contribution of the  $i$ th component of  $\mathbf{u}$  should be scaled according to its relevance to  $\mathbf{v}$ , and vice versa. In effect, this is what model (6) achieves.

As a result of the assumption of symmetry, both these models are ‘bag of words’ models and word order insensitive. Relaxing the assumption of symmetry in the case of the simple additive model produces a model which weighs the contribution of the two components differently:

$$p_i = \alpha u_i + \beta v_i \quad (7)$$

This allows additive models to become more syntax aware, since semantically important constituents can participate more actively in the composition. As an example if we set  $\alpha$  to 0.4 and  $\beta$  to 0.6, then  $\mathbf{horse} = [0 \ 2.4 \ 0.8 \ 4 \ 1.6]$  and  $\mathbf{run} = [0.6 \ 4.8 \ 2.4 \ 2.4 \ 0]$ , and their sum  $\mathbf{horse} + \mathbf{run} = [0.6 \ 5.6 \ 3.2 \ 6.4 \ 1.6]$ .

An extreme form of this differential in the contribution of constituents is where one of the vectors, say  $\mathbf{u}$ , contributes nothing at all to the combination:

$$p_i = v_j \quad (8)$$

Admittedly the model in (8) is impoverished and rather simplistic, however it can serve as a simple baseline against which to compare more sophisticated models.

The models considered so far assume that components do not ‘interfere’ with each other, i.e., that

only the  $i$ th components of  $\mathbf{u}$  and  $\mathbf{v}$  contribute to the  $i$ th component of  $\mathbf{p}$ . Another class of models can be derived by relaxing this constraint. To give a concrete example, circular convolution is an instance of the general multiplicative model which breaks this constraint by allowing  $u_j$  to contribute to  $p_i$ :

$$p_i = \sum_j u_j \cdot v_{i-j} \quad (9)$$

It is also possible to re-introduce the dependence on  $K$  into the model of vector composition. For additive models, a natural way to achieve this is to include further vectors into the summation. These vectors are not arbitrary and ideally they must exhibit some relation to the words of the construction under consideration. When modeling predicate-argument structures, Kintsch (2001) proposes including one or more distributional neighbors,  $\mathbf{n}$ , of the predicate:

$$\mathbf{p} = \mathbf{u} + \mathbf{v} + \sum \mathbf{n} \quad (10)$$

Note that considerable latitude is allowed in selecting the appropriate neighbors. Kintsch (2001) considers only the  $m$  most similar neighbors to the predicate, from which he subsequently selects  $k$ , those most similar to its argument. So, if in the composition of *horse* with *run*, the chosen neighbor is *ride*,  $\mathbf{ride} = [2 \ 15 \ 7 \ 9 \ 1]$ , then this produces the representation  $\mathbf{horse} + \mathbf{run} + \mathbf{ride} = [3 \ 29 \ 13 \ 23 \ 5]$ . In contrast to the simple additive model, this extended model is sensitive to syntactic structure, since  $\mathbf{n}$  is chosen from among the neighbors of the predicate, distinguishing it from the argument.

Although we have presented multiplicative and additive models separately, there is nothing inherent in our formulation that disallows their combination. The proposal is not merely notational. One potential drawback of multiplicative models is the effect of components with value zero. Since the product of zero with any number is itself zero, the presence of zeroes in either of the vectors leads to information being essentially thrown away. Combining the multiplicative model with an additive model, which does not suffer from this problem, could mitigate this problem:

$$p_i = \alpha u_i + \beta v_i + \gamma u_i v_i \quad (11)$$

where  $\alpha$ ,  $\beta$ , and  $\gamma$  are weighting constants.

## 4 Evaluation Set-up

We evaluated the models presented in Section 3 on a sentence similarity task initially proposed by Kintsch (2001). In his study, Kintsch builds a model of how a verb's meaning is modified in the context of its subject. He argues that the subjects of *ran* in *The color ran* and *The horse ran* select different senses of *ran*. This change in the verb's sense is equated to a shift in its position in semantic space. To quantify this shift, Kintsch proposes measuring similarity relative to other verbs acting as landmarks, for example *gallop* and *dissolve*. The idea here is that an appropriate composition model when applied to *horse* and *ran* will yield a vector closer to the landmark *gallop* than *dissolve*. Conversely, when *color* is combined with *ran*, the resulting vector will be closer to *dissolve* than *gallop*.

Focusing on a single compositional structure, namely intransitive verbs and their subjects, is a good point of departure for studying vector combination. Any adequate model of composition must be able to represent argument-verb meaning. Moreover by using a minimal structure we factor out inessential degrees of freedom and are able to assess the merits of different models on an equal footing. Unfortunately, Kintsch (2001) demonstrates how his own composition algorithm works intuitively on a few hand selected examples but does not provide a comprehensive test set. In order to establish an independent measure of sentence similarity, we assembled a set of experimental materials and elicited similarity ratings from human subjects. In the following we describe our data collection procedure and give details on how our composition models were constructed and evaluated.

**Materials and Design** Our materials consisted of sentences with an intransitive verb and its subject. We first compiled a list of intransitive verbs from CELEX<sup>2</sup>. All occurrences of these verbs with a subject noun were next extracted from a RASP parsed (Briscoe and Carroll, 2002) version of the British National Corpus (BNC). Verbs and nouns that were attested less than fifty times in the BNC were removed as they would result in unreliable vectors. Each reference subject-verb tuple (e.g., *horse ran*) was paired with two landmarks, each a synonym of the verb. The landmarks were chosen so as to represent distinct verb senses, one compatible

with the reference (e.g., *horse galloped*) and one incompatible (e.g., *horse dissolved*). Landmarks were taken from WordNet (Fellbaum, 1998). Specifically, they belonged to different synsets and were maximally dissimilar as measured by the Jiang and Conrath (1997) measure.<sup>3</sup>

Our initial set of candidate materials consisted of 20 verbs, each paired with 10 nouns, and 2 landmarks (400 pairs of sentences in total). These were further pretested to allow the selection of a subset of items showing clear variations in sense as we wanted to have a balanced set of similar and dissimilar sentences. In the pretest, subjects saw a reference sentence containing a subject-verb tuple and its landmarks and were asked to choose which landmark was most similar to the reference or neither. Our items were converted into simple sentences (all in past tense) by adding articles where appropriate. The stimuli were administered to four separate groups; each group saw one set of 100 sentences. The pretest was completed by 53 participants.

For each reference verb, the subjects' responses were entered into a contingency table, whose rows corresponded to nouns and columns to each possible answer (i.e., one of the two landmarks). Each cell recorded the number of times our subjects selected the landmark as compatible with the noun or not. We used Fisher's exact test to determine which verbs and nouns showed the greatest variation in landmark preference and items with *p*-values greater than 0.001 were discarded. This yielded a reduced set of experimental items (120 in total) consisting of 15 reference verbs, each with 4 nouns, and 2 landmarks.

**Procedure and Subjects** Participants first saw a set of instructions that explained the sentence similarity task and provided several examples. Then the experimental items were presented; each contained two sentences, one with the reference verb and one with its landmark. Examples of our items are given in Table 1. Here, *burn* is a high similarity landmark (High) for the reference *The fire glowed*, whereas *beam* is a low similarity landmark (Low). The opposite is the case for the reference *The face*

<sup>3</sup>We assessed a wide range of semantic similarity measures using the WordNet similarity package (Pedersen et al., 2004). Most of them yielded similar results. We selected Jiang and Conrath's measure since it has been shown to perform consistently well across several cognitive and NLP tasks (Budanitsky and Hirst, 2001).

<sup>2</sup><http://www.ru.nl/celex/>

Noun	Reference	High	Low
The fire	glowed	burned	beamed
The face	glowed	beamed	burned
The child	strayed	roamed	digressed
The discussion	strayed	digressed	roamed
The sales	slumped	declined	slouched
The shoulders	slumped	slouched	declined

Table 1: Example Stimuli with High and Low similarity landmarks

*glowed*. Sentence pairs were presented serially in random order. Participants were asked to rate how similar the two sentences were on a scale of one to seven. The study was conducted remotely over the Internet using Webexp<sup>4</sup>, a software package designed for conducting psycholinguistic studies over the web. 49 unpaid volunteers completed the experiment, all native speakers of English.

**Analysis of Similarity Ratings** The reliability of the collected judgments is important for our evaluation experiments; we therefore performed several tests to validate the quality of the ratings. First, we examined whether participants gave high ratings to high similarity sentence pairs and low ratings to low similarity ones. Figure 2 presents a box-and-whisker plot of the distribution of the ratings. As we can see sentences with high similarity landmarks are perceived as more similar to the reference sentence. A Wilcoxon rank sum test confirmed that the difference is statistically significant ( $p < 0.01$ ). We also measured how well humans agree in their ratings. We employed leave-one-out resampling (Weiss and Kulikowski, 1991), by correlating the data obtained from each participant with the ratings obtained from all other participants. We used Spearman’s  $\rho$ , a non parametric correlation coefficient, to avoid making any assumptions about the distribution of the similarity ratings. The average inter-subject agreement<sup>5</sup> was  $\rho = 0.40$ . We believe that this level of agreement is satisfactory given that naive subjects are asked to provide judgments on fine-grained semantic distinctions (see Table 1). More evidence that this is not an easy task comes from Figure 2 where we observe some overlap in the ratings for High and Low similarity items.

<sup>4</sup><http://www.webexp.info/>

<sup>5</sup>Note that Spearman’s rho tends to yield lower coefficients compared to parametric alternatives such as Pearson’s  $r$ .

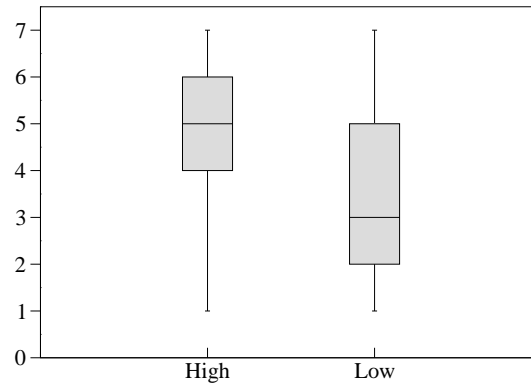


Figure 2: Distribution of elicited ratings for High and Low similarity items

**Model Parameters** Irrespectively of their form, all composition models discussed here are based on a semantic space for representing the meanings of individual words. The semantic space we used in our experiments was built on a lemmatised version of the BNC. Following previous work (Bullinaria and Levy, 2007), we optimized its parameters on a word-based semantic similarity task. The task involves examining the degree of linear relationship between the human judgments for two individual words and vector-based similarity values. We experimented with a variety of dimensions (ranging from 50 to 500,000), vector component definitions (e.g., pointwise mutual information or log likelihood ratio) and similarity measures (e.g., cosine or confusion probability). We used WordSim353, a benchmark dataset (Finkelstein et al., 2002), consisting of relatedness judgments (on a scale of 0 to 10) for 353 word pairs.

We obtained best results with a model using a context window of five words on either side of the target word, the cosine measure, and 2,000 vector components. The latter were the most common context words (excluding a stop list of function words). These components were set to the ratio of the probability of the context word given the target word to the probability of the context word overall. This configuration gave high correlations with the WordSim353 similarity judgments using the cosine measure. In addition, Bullinaria and Levy (2007) found that these parameters perform well on a number of other tasks such as the synonymy task from the *Test of English as a Foreign Language* (TOEFL).

Our composition models have no additional pa-

rameters beyond the semantic space just described, with three exceptions. First, the additive model in (7) weighs differentially the contribution of the two constituents. In our case, these are the subject noun and the intransitive verb. To this end, we optimized the weights on a small held-out set. Specifically, we considered eleven models, varying in their weightings, in steps of 10%, from 100% noun through 50% of both verb and noun to 100% verb. For the best performing model the weight for the verb was 80% and for the noun 20%. Secondly, we optimized the weightings in the combined model (11) with a similar grid search over its three parameters. This yielded a weighted sum consisting of 95% verb, 0% noun and 5% of their multiplicative combination. Finally, Kintsch’s (2001) additive model has two extra parameters. The  $m$  neighbors most similar to the predicate, and the  $k$  of  $m$  neighbors closest to its argument. In our experiments we selected parameters that Kintsch reports as optimal. Specifically,  $m$  was set to 20 and  $k$  to 1.

**Evaluation Methodology** We evaluated the proposed composition models in two ways. First, we used the models to estimate the cosine similarity between the reference sentence and its landmarks. We expect better models to yield a pattern of similarity scores like those observed in the human ratings (see Figure 2). A more scrupulous evaluation requires directly correlating all the individual participants’ similarity judgments with those of the models.<sup>6</sup> We used Spearman’s  $\rho$  for our correlation analyses. Again, better models should correlate better with the experimental data. We assume that the inter-subject agreement can serve as an upper bound for comparing the fit of our models against the human judgments.

## 5 Results

Our experiments assessed the performance of seven composition models. These included three additive models, i.e., simple addition (equation (5), Add), weighted addition (equation (7), WeightAdd), and Kintsch’s (2001) model (equation (10), Kintsch), a multiplicative model (equation (6), Multiply), and also a model which combines multiplication with

<sup>6</sup>We avoided correlating the model predictions with averaged participant judgments as this is inappropriate given the ordinal nature of the scale of these judgments and also leads to a dependence between the number of participants and the magnitude of the correlation coefficient.

Model	High	Low	$\rho$
NonComp	0.27	0.26	0.08**
Add	0.59	0.59	0.04*
WeightAdd	0.35	0.34	0.09**
Kintsch	0.47	0.45	0.09**
Multiply	0.42	0.28	0.17**
Combined	0.38	0.28	0.19**
UpperBound	4.94	3.25	0.40**

Table 2: Model means for High and Low similarity items and correlation coefficients with human judgments (\*:  $p < 0.05$ , \*\*:  $p < 0.01$ )

addition (equation (11), Combined). As a baseline we simply estimated the similarity between the reference verb and its landmarks without taking the subject noun into account (equation (8), NonComp). Table 2 shows the average model ratings for High and Low similarity items. For comparison, we also show the human ratings for these items (UpperBound). Here, we are interested in relative differences, since the two types of ratings correspond to different scales. Model similarities have been estimated using cosine which ranges from 0 to 1, whereas our subjects rated the sentences on a scale from 1 to 7.

The simple additive model fails to distinguish between High and Low Similarity items. We observe a similar pattern for the non compositional baseline model, the weighted additive model and Kintsch (2001). The multiplicative and combined models yield means closer to the human ratings. The difference between High and Low similarity values estimated by these models are statistically significant ( $p < 0.01$  using the Wilcoxon rank sum test). Figure 3 shows the distribution of estimated similarities under the multiplicative model.

The results of our correlation analysis are also given in Table 2. As can be seen, all models are significantly correlated with the human ratings. In order to establish which ones fit our data better, we examined whether the correlation coefficients achieved differ significantly using a  $t$ -test (Cohen and Cohen, 1983). The lowest correlation ( $\rho = 0.04$ ) is observed for the simple additive model which is not significantly different from the non-compositional baseline model. The weighted additive model ( $\rho = 0.09$ ) is not significantly different from the baseline either or Kintsch (2001) ( $\rho = 0.09$ ). Given that the basis

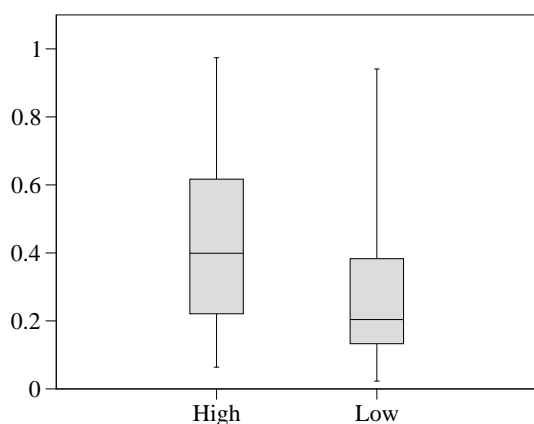


Figure 3: Distribution of predicted similarities for the vector multiplication model on High and Low similarity items

of Kintsch’s model is the summation of the verb, a neighbor close to the verb and the noun, it is not surprising that it produces results similar to a summation which weights the verb more heavily than the noun. The multiplicative model yields a better fit with the experimental data,  $\rho = 0.17$ . The combined model is best overall with  $\rho = 0.19$ . However, the difference between the two models is not statistically significant. Also note that in contrast to the combined model, the multiplicative model does not have any free parameters and hence does not require optimization for this particular task.

## 6 Discussion

In this paper we presented a general framework for vector-based semantic composition. We formulated composition as a function of two vectors and introduced several models based on addition and multiplication. Despite the popularity of additive models, our experimental results showed the superiority of models utilizing multiplicative combinations, at least for the sentence similarity task attempted here. We conjecture that the additive models are not sensitive to the fine-grained meaning distinctions involved in our materials. Previous applications of vector addition to document indexing (Deerwester et al., 1990) or essay grading (Landauer et al., 1997) were more concerned with modeling the gist of a document rather than the meaning of its sentences. Importantly, additive models capture composition by considering *all* vector components representing the meaning of the verb and its subject,

whereas multiplicative models consider a subset, namely non-zero components. The resulting vector is sparser but expresses more succinctly the meaning of the predicate-argument structure, and thus allows semantic similarity to be modelled more accurately.

Further research is needed to gain a deeper understanding of vector composition, both in terms of modeling a wider range of structures (e.g., adjective-noun, noun-noun) and also in terms of exploring the space of models more fully. We anticipate that more substantial correlations can be achieved by implementing more sophisticated models from within the framework outlined here. In particular, the general class of multiplicative models (see equation (4)) appears to be a fruitful area to explore. Future directions include constraining the number of free parameters in linguistically plausible ways and scaling to larger datasets.

The applications of the framework discussed here are many and varied both for cognitive science and NLP. We intend to assess the potential of our composition models on context sensitive semantic priming (Till et al., 1988) and inductive inference (Heit and Rubinstein, 1994). NLP tasks that could benefit from composition models include paraphrase identification and context-dependent language modeling (Coccaro and Jurafsky, 1998).

## References

- E. Briscoe, J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, 1499–1504, Las Palmas, Canary Islands.
- A. Budanitsky, G. Hirst. 2001. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Proceedings of ACL Workshop on WordNet and Other Lexical Resources*, Pittsburgh, PA.
- J. Bullinaria, J. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.
- F. Choi, P. Wiemer-Hastings, J. Moore. 2001. Latent semantic analysis for text segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 109–117, Pittsburgh, PA.
- N. Coccaro, D. Jurafsky. 1998. Towards better integration of semantic predictors in statistical language modeling. In *Proceedings of the 5th International Conference on Spoken Language Processing*, Sydney, Australia.

- J. Cohen, P. Cohen. 1983. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Hillsdale, NJ: Erlbaum.
- S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, R. A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.
- G. Denhire, B. Lemaire. 2004. A computational model of children’s semantic memory. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*, 297–302, Chicago, IL.
- C. Fellbaum, ed. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.
- L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, E. Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.
- J. Fodor, Z. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71.
- P. W. Foltz, W. Kintsch, T. K. Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Process*, 15:285–307.
- S. Frank, M. Koppen, L. Noordman, W. Vonk. 2007. World knowledge in computational models of discourse comprehension. *Discourse Processes*. In press.
- G. Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers.
- Z. Harris. 1968. *Mathematical Structures of Language*. Wiley, New York.
- E. Heit, J. Rubinstein. 1994. Similarity and property effects in inductive reasoning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20:411–422.
- J. J. Jiang, D. W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research in Computational Linguistics*, Taiwan.
- W. Kintsch. 2001. Predication. *Cognitive Science*, 25(2):173–202.
- T. K. Landauer, S. T. Dumais. 1997. A solution to Plato’s problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.
- T. K. Landauer, D. Laham, B. Rehder, M. E. Schreiner. 1997. How well can passage meaning be derived without using word order: A comparison of latent semantic analysis and humans. In *Proceedings of 19th Annual Conference of the Cognitive Science Society*, 412–417, Stanford, CA.
- K. Lund, C. Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments & Computers*, 28:203–208.
- D. McCarthy, R. Koeling, J. Weeds, J. Carroll. 2004. Finding predominant senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 280–287, Barcelona, Spain.
- S. McDonald. 2000. *Environmental Determinants of Lexical Processing Effort*. Ph.D. thesis, University of Edinburgh.
- R. Montague. 1974. English as a formal language. In R. Montague, ed., *Formal Philosophy*. Yale University Press, New Haven, CT.
- H. Neville, J. L. Nichol, A. Barss, K. I. Forster, M. F. Garrett. 1991. Syntactically based sentence processing classes: evidence from event-related brain potentials. *Journal of Cognitive Neuroscience*, 3:151–165.
- S. Padó, M. Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.
- T. Pedersen, S. Patwardhan, J. Michelizzi. 2004. WordNet::similarity - measuring the relatedness of concepts. In *Proceedings of the 5th Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, 38–41, Boston, MA.
- T. A. Plate. 1991. Holographic reduced representations: Convolution algebra for compositional distributed representations. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, 30–35, Sydney, Australia.
- G. Salton, A. Wong, C. S. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.
- P. Schone, D. Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 100–108, Pittsburgh, PA.
- H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.
- P. Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.
- R. E. Till, E. F. Mross, W. Kintsch. 1988. Time course of priming for associate and inference words in discourse context. *Memory and Cognition*, 16:283–299.
- S. M. Weiss, C. A. Kulikowski. 1991. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann, San Mateo, CA.
- R. F. West, K. E. Stanovich. 1986. Robust effects of syntactic structure on visual word processing. *Journal of Memory and Cognition*, 14:104–112.



# Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition

Andrew Arnold, Ramesh Nallapati and William W. Cohen

Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA  
{aarnold, nmramesh, wcohen}@cs.cmu.edu

## Abstract

We present a novel hierarchical prior structure for supervised transfer learning in named entity recognition, motivated by the common structure of feature spaces for this task across natural language data sets. The problem of transfer learning, where information gained in one learning task is used to improve performance in another related task, is an important new area of research. In the subproblem of domain adaptation, a model trained over a source domain is generalized to perform well on a related target domain, where the two domains' data are distributed similarly, but not identically. We introduce the concept of groups of closely-related domains, called *genres*, and show how inter-genre adaptation is related to domain adaptation. We also examine multi-task learning, where two domains may be related, but where the concept to be learned in each case is distinct. We show that our prior conveys useful information across domains, genres and tasks, while remaining robust to spurious signals not related to the target domain and concept. We further show that our model generalizes a class of similar hierarchical priors, smoothed to varying degrees, and lay the groundwork for future exploration in this area.

## 1 Introduction

### 1.1 Problem definition

Consider the task of *named entity recognition* (NER). Specifically, you are given a corpus of news articles in which all tokens have been labeled as either belonging to personal name mentions or not. The standard supervised machine learning problem is to learn a classifier over this training data that will successfully label unseen test data drawn from the same distribution as the training data, where “same distribution” could mean anything from having the train and test articles written by the same author to

having them written in the same language. Having successfully trained a named entity classifier on this news data, now consider the problem of learning to classify tokens as names in e-mail data. An intuitive solution might be to simply retrain the classifier, *de novo*, on the e-mail data. Practically, however, large, labeled datasets are often expensive to build and this solution would not scale across a large number of different datasets.

Clearly the problems of identifying names in news articles and e-mails are closely related, and learning to do well on one should help your performance on the other. At the same time, however, there are serious differences between the two problems that need to be addressed. For instance, capitalization, which will certainly be a useful feature in the news problem, may prove less informative in the e-mail data since the rules of capitalization are followed less strictly in that domain.

These are the problems we address in this paper. In particular, we develop a novel prior for named entity recognition that exploits the hierarchical feature space often found in natural language domains (§1.2) and allows for the transfer of information from labeled datasets in other domains (§1.3). §2 introduces the *maximum entropy* (maxent) and *conditional random field* (CRF) learning techniques employed, along with specifications for the design and training of our hierarchical prior. Finally, in §3 we present an empirical investigation of our prior's performance against a number of baselines, demonstrating both its effectiveness and robustness.

### 1.2 Hierarchical feature trees

In many NER problems, features are often constructed as a series of transformations of the input training data, performed in sequence. Thus, if our task is to identify tokens as either being (*O*)*outside* or (*I*)*inside* person names, and we are given the labeled

sample training sentence:

*O O O O O I*  
 Give the book to Professor Caldwell  
 (1)

one such useful feature might be: *Is the token one slot to the left of the current token Professor?* We can represent this symbolically as *L.I.Professor* where we describe the whole space of useful features of this form as:  $\{direction = (L)eft, (C)urrent, (R)ight\} \cdot \{distance = 1, 2, 3, \dots\} \cdot \{value = Professor, book, \dots\}$ . We can conceptualize this structure as a tree, where each slot in the symbolic name of a feature is a branch and each period between slots represents another level, going from root to leaf as read left to right. Thus a subsection of the entire feature tree for the token *Caldwell* could be drawn as in Figure 1 (zoomed in on the section of the tree where the *L.I.Professor* feature resides).

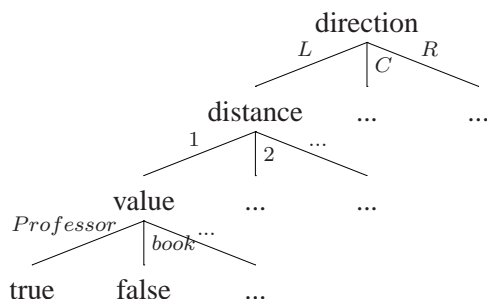


Figure 1: Graphical representation of a hierarchical feature tree for token *Caldwell* in example Sentence 1.

Representing feature spaces with this kind of tree, besides often coinciding with the explicit language used by common natural language toolkits (Cohen, 2004), has the added benefit of allowing a model to easily back-off, or smooth, to decreasing levels of specificity. For example, the leaf level of the feature tree for our sample Sentence 1 tells us that the word *Professor* is important, with respect to labeling person names, when located one slot to the left of the current word being classified. This may be useful in the context of an academic corpus, but might be less useful in a medical domain where the word *Professor* occurs less often. Instead, we might want to learn the related feature *L.I.Dr*. In fact, it might be useful to generalize across multiple domains the fact that the word immediately preceding the current word is often important with respect

---

LeftToken.*
LeftToken.IsWord.*
LeftToken.IsWord.IsTitle.*
LeftToken.IsWord.IsTitle.equals.*
LeftToken.IsWord.IsTitle.equals.mr

---

Table 1: A few examples of the feature hierarchy

to the named entity status of the current word. This is easily accomplished by backing up one level from a leaf in the tree structure to its parent, to represent a class of features such as *L.I.\**. It has been shown empirically that, while the significance of *particular* features might vary between domains and tasks, certain generalized *classes* of features retain their importance across domains (Minkov et al., 2005). By backing-off in this way, we can use the feature hierarchy as a prior for transferring beliefs about the significance of entire *classes* of features across domains and tasks. Some examples illustrating this idea are shown in table 1.

### 1.3 Transfer learning

When only the type of data being examined is allowed to vary (from news articles to e-mails, for example), the problem is called *domain adaptation* (Daumé III and Marcu, 2006). When the task being learned varies (say, from identifying person names to identifying protein names), the problem is called *multi-task learning* (Caruana, 1997). Both of these are considered specific types of the overarching *transfer learning* problem, and both seem to require a way of altering the classifier learned on the first problem (called the *source domain*, or *source task*) to fit the specifics of the second problem (called the *target domain*, or *target task*).

More formally, given an example  $x$  and a class label  $y$ , the standard statistical classification task is to assign a probability,  $p(y|x)$ , to  $x$  of belonging to class  $y$ . In the binary classification case the labels are  $Y \in \{0, 1\}$ . In the case we examine, each example  $x_i$  is represented as a vector of binary features  $(f_1(x_i), \dots, f_F(x_i))$  where  $F$  is the number of features. The data consists of two disjoint subsets: the training set  $(X_{train}, Y_{train}) = \{(x_1, y_1) \dots (x_N, y_N)\}$ , available to the model for its training and the test set  $X_{test} = (x_1, \dots, x_M)$ , upon which we want to use our trained classifier to make predictions.

In the paradigm of *inductive learning*,  $(X_{train}, Y_{train})$  are known, while both  $X_{test}$  and  $Y_{test}$  are completely hidden during training time. In this cases  $X_{test}$  and  $X_{train}$  are both assumed to have been drawn from the same distribution,  $\mathcal{D}$ . In the setting of *transfer learning*, however, we would like to apply our trained classifier to examples drawn from a distribution different from the one upon which it was trained. We therefore assume there are two different distributions,  $\mathcal{D}^{source}$  and  $\mathcal{D}^{target}$ , from which data may be drawn. Given this notation we can then precisely state the transfer learning problem as trying to assign labels  $Y_{test}^{target}$  to test data  $X_{test}^{target}$  drawn from  $\mathcal{D}^{target}$ , given training data  $(X_{train}^{source}, Y_{train}^{source})$  drawn from  $\mathcal{D}^{source}$ .

In this paper we focus on two subproblems of transfer learning:

- *domain adaptation*, where we assume  $Y$  (the set of possible labels) is the same for both  $\mathcal{D}^{source}$  and  $\mathcal{D}^{target}$ , while  $\mathcal{D}^{source}$  and  $\mathcal{D}^{target}$  themselves are allowed to vary between domains.
- *multi-task learning* (Ando and Zhang, 2005; Caruana, 1997; Sutton and McCallum, 2005; Zhang et al., 2005) in which the task (and label set) is allowed to vary from source to target.

Domain adaptation can be further distinguished by the degree of relatedness between the source and target domains. For example, in this work we group data collected in the same medium (e.g., all annotated e-mails or all annotated news articles) as belonging to the same *genre*. Although the specific boundary between domain and genre for a particular set of data is often subjective, it is nevertheless a useful distinction to draw.

One common way of addressing the transfer learning problem is to use a *prior* which, in conjunction with a probabilistic model, allows one to specify *a priori* beliefs about a distribution, thus biasing the results a learning algorithm would have produced had it only been allowed to see the training data (Raina et al., 2006). In the example from §1.1, our belief that capitalization is less strict in e-mails than in news articles could be encoded in a prior that biased the importance of the capitalization feature to be lower for e-mails than news articles. In the next section we address the problem of how to come up with a suitable prior for transfer learning across named entity recognition problems.

## 2 Models considered

### 2.1 Basic Conditional Random Fields

In this work, we will base our work on Conditional Random Fields (CRF's) (Lafferty et al., 2001), which are now one of the most preferred sequential models for many natural language processing tasks.

The parametric form of the CRF for a sentence of length  $n$  is given as follows:

$$p_{\Lambda}(\mathbf{Y} = \mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i=1}^n \sum_{j=1}^F f_j(\mathbf{x}, y_i) \lambda_j\right) \quad (2)$$

where  $Z(\mathbf{x})$  is the normalization term. CRF learns a model consisting of a set of weights  $\Lambda = \{\lambda_1 \dots \lambda_F\}$  over the features so as to maximize the conditional likelihood of the training data,  $p(Y_{train}|X_{train})$ , given the model  $p_{\Lambda}$ .

### 2.2 CRF with Gaussian priors

To avoid overfitting the training data, these  $\lambda$ 's are often further constrained by the use of a Gaussian prior (Chen and Rosenfeld, 1999) with diagonal covariance,  $\mathcal{N}(\mu, \sigma^2)$ , which tries to maximize:

$$\operatorname{argmax}_{\Lambda} \sum_{k=1}^N \left( \log p_{\Lambda}(\mathbf{y}_k|\mathbf{x}_k) \right) - \beta \sum_j^F \frac{(\lambda_j - \mu_j)^2}{2\sigma_j^2}$$

where  $\beta > 0$  is a parameter controlling the amount of regularization, and  $N$  is the number of sentences in the training set.

### 2.3 Source trained priors

One recently proposed method (Chelba and Acero, 2004) for transfer learning in Maximum Entropy models<sup>1</sup> involves modifying the  $\mu$ 's of this Gaussian prior. First a model of the source domain,  $\Lambda^{source}$ , is learned by training on  $\{X_{train}^{source}, Y_{train}^{source}\}$ . Then a model of the target domain is trained over a limited set of labeled target data  $\{X_{train}^{target}, Y_{train}^{target}\}$ , but instead of regularizing this  $\Lambda^{target}$  to be near zero (i.e. setting  $\mu = 0$ ),  $\Lambda^{target}$  is instead regularized towards the previously learned source values  $\Lambda^{source}$  (by setting  $\mu = \Lambda^{source}$ , while  $\sigma^2$  remains 1) and thus minimizing  $(\Lambda^{target} - \Lambda^{source})^2$ .

<sup>1</sup>Maximum Entropy models are special cases of CRFs that use the I.I.D. assumption. The method under discussion can also be extended to CRF directly.

Note that, since this model requires  $Y_{train}^{target}$  in order to learn  $\Lambda^{target}$ , it, in effect, requires two distinct labeled training datasets: one on which to *train* the prior, and another on which to learn the model’s final weights (which we call *tuning*), using the previously trained prior for regularization. If we are unable to find a match between features in the training and tuning datasets (for instance, if a word appears in the tuning corpus but not the training), we back-off to a standard  $\mathcal{N}(0, 1)$  prior for that feature.

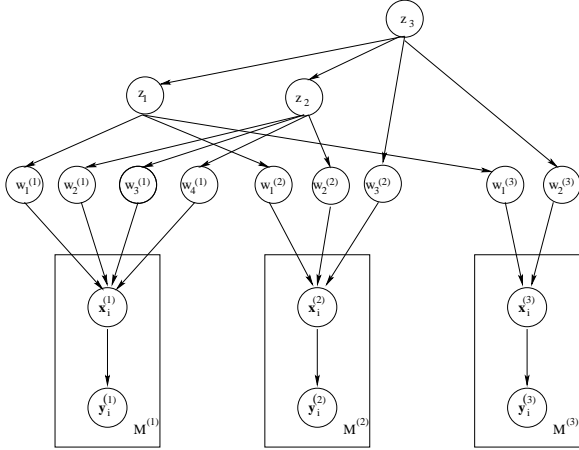


Figure 2: Graphical representation of the hierarchical transfer model.

## 2.4 New model: Hierarchical prior model

In this section, we will present a new model that learns simultaneously from multiple domains, by taking advantage of our feature hierarchy.

We will assume that there are  $D$  domains in which we are learning simultaneously. Let there be  $M_d$  training data in each domain  $d$ . For our experiments with non-identically distributed, independent data, we use conditional random fields (cf. §2.1). However, this model can be extended to any discriminative probabilistic model such as the MaxEnt model. Let  $\Lambda^{(d)} = (\lambda_1^{(d)}, \dots, \lambda_{F_d}^{(d)})$  be the parameters of the discriminative model in the domain  $d$  where  $F_d$  represents the number of features in the domain  $d$ .

Further, we will also assume that the features of different domains share a common hierarchy represented by a tree  $\mathcal{T}$ , whose leaf nodes are the features themselves (cf. Figure 1). The model parameters  $\Lambda^{(d)}$ , then, form the parameters of the leaves of this hierarchy. Each non-leaf node  $n \in \text{non-leaf}(\mathcal{T})$  of

the tree is also associated with a hyper-parameter  $z_n$ . Note that since the hierarchy is a tree, each node  $n$  has only one parent, represented by  $\text{pa}(n)$ . Similarly, we represent the set of children nodes of a node  $n$  as  $\text{ch}(n)$ .

The entire graphical model for an example consisting of three domains is shown in Figure 2. The conditional likelihood of the entire training data  $(\mathbf{y}, \mathbf{x}) = \{(\mathbf{y}_1^{(d)}, \mathbf{x}_1^{(d)}), \dots, (\mathbf{y}_{M_d}^{(d)}, \mathbf{x}_{M_d}^{(d)})\}_{d=1}^D$  is given by:

$$\begin{aligned}
 P(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathbf{z}) &= \left\{ \prod_{d=1}^D \prod_{k=1}^{M_d} P(\mathbf{y}_k^{(d)}|\mathbf{x}_k^{(d)}, \Lambda^{(d)}) \right\} \\
 &\times \left\{ \prod_{d=1}^D \prod_{f=1}^{F_d} \mathcal{N}(\lambda_f^{(d)}|z_{\text{pa}(f^{(d)})}, 1) \right\} \\
 &\times \left\{ \prod_{n \in \mathcal{T}_{\text{nonleaf}}} \mathcal{N}(z_n|z_{\text{pa}(n)}, 1) \right\} \quad (3)
 \end{aligned}$$

where the terms in the first line of eq. (3) represent the likelihood of data in each domain given their corresponding model parameters, the second line represents the likelihood of each model parameter in each domain given the hyper-parameter of its parent in the tree hierarchy of features and the last term goes over the entire tree  $\mathcal{T}$  except the leaf nodes. Note that in the last term, the hyper-parameters are shared across the domains, so there is no product over  $d$ .

We perform a MAP estimation for each model parameter as well as the hyper-parameters. Accordingly, the estimates are given as follows:

$$\begin{aligned}
 \lambda_f^{(d)} &= \sum_{i=1}^{M_d} \frac{\partial}{\partial \lambda_f^{(d)}} \left( \log P(\mathbf{y}_i^d|\mathbf{x}_i^{(d)}, \Lambda^{(d)}) \right) \\
 &+ z_{\text{pa}(f^{(d)})} \\
 z_n &= \frac{z_{\text{pa}(n)} + \sum_{i \in \text{ch}(n)} (\lambda|z)_i}{1 + |\text{ch}(n)|} \quad (4)
 \end{aligned}$$

where we used the notation  $(\lambda|z)_i$  because node  $i$ , the child node of  $n$ , could be a parameter node or a hyper-parameter node depending on the position of the node  $n$  in the hierarchy. Essentially, in this model, the weights of the leaf nodes (model parameters) depend on the log-likelihood as well as the prior weight of its parent. Additionally, the weight

of each hyper-parameter node in the tree is computed as the average of all its children nodes and its parent, resulting in a *smoothing* effect, both up and down the tree.

## 2.5 An approximate Hierarchical prior model

The Hierarchical prior model is a theoretically well founded model for transfer learning through feature heirarchy. However, our preliminary experiments indicated that its performance on real-life data sets is not as good as expected. Although a more thorough investigation needs to be carried out, our analysis indicates that the main reason for this phenomenon is over-smoothing. In other words, by letting the information propagate from the leaf nodes in the hierarchy all the way to the root node, the model loses its ability to discriminate between its features.

As a solution to this problem, we propose an approximate version of this model that weds ideas from the exact heirarchical prior model and the Chelba model.

As with the Chelba prior method in §2.3, this approximate hierarchical method also requires two distinct data sets, one for training the prior and another for tuning the final weights. Unlike Chelba, we smooth the weights of the priors using the feature-tree hierarchy presented in §1.1, like the hierarchical prior model.

For smoothing of each feature weight, we chose to back-off in the tree as little as possible until we had a large enough sample of prior data (measured as  $M$ , the number of subtrees below the current node) on which to form a reliable estimate of the mean and variance of each feature or class of features. For example, if the tuning data set is as in Sentence 1, but the prior contains no instances of the word *Professor*, then we would back-off and compute the prior mean and variance on the next higher level in the tree. Thus the prior for *L.I.Professor* would be  $\mathcal{N}(\text{mean}(L.I.*), \text{variance}(L.I.*))$ , where  $\text{mean}()$  and  $\text{variance}()$  of *L.I.\** are the sample mean and variance of all the features in the prior dataset that match the pattern *L.I.\** – or, put another way, all the siblings of *L.I.Professor* in the feature tree. If fewer than  $M$  such siblings exist, we continue backing-off, up the tree, until an ancestor with sufficient descendants is found. A detailed description of the approximate hierarchical algorithm is shown in table 2.

---

**Input:**  $\mathcal{D}^{source} = (X_{train}^{source}, Y_{train}^{source})$   
 $\mathcal{D}^{target} = (X_{train}^{target}, Y_{train}^{target})$ ;  
 Feature sets  $\mathcal{F}^{source}, \mathcal{F}^{target}$ ;  
 Feature Hierarchies  $\mathcal{H}^{source}, \mathcal{H}^{target}$   
 Minimum membership size  $M$

---

Train CRF using  $\mathcal{D}^{source}$  to obtain feature weights  $\Lambda^{source}$

For each feature  $f \in \mathcal{F}^{target}$

  Initialize: node  $n = f$

  While ( $n \notin \mathcal{H}^{source}$

  or  $|\text{Leaves}(\mathcal{H}^{source}(n))| \leq M$ )

  and  $n \neq \text{root}(\mathcal{H}^{target})$

$n \leftarrow \text{Pa}(\mathcal{H}^{target}(n))$

  Compute  $\mu_f$  and  $\sigma_f$  using the sample

$\{\lambda_i^{source} \mid i \in \text{Leaves}(\mathcal{H}^{source}(n))\}$

Train Gaussian prior CRF using  $\mathcal{D}^{target}$  as data and  $\{\mu_f\}$  and  $\{\sigma_f\}$  as Gaussian prior parameters.

**Output:** Parameters of the new CRF  $\Lambda^{target}$ .

---

Table 2: Algorithm for approximate hierarchical prior:  $\text{Pa}(\mathcal{H}^{source}(n))$  is the parent of node  $n$  in feature hierarchy  $\mathcal{H}^{source}$ ;  $|\text{Leaves}(\mathcal{H}^{source}(n))|$  indicates the number of leaf nodes (basic features) under a node  $n$  in the hierarchy  $\mathcal{H}^{source}$ .

It is important to note that this smoothed tree is an approximation of the exact model presented in §2.4 and thus an important parameter of this method in practice is the degree to which one chooses to smooth up or down the tree. One of the benefits of this model is that the semantics of the hierarchy (how to define a feature, a parent, how and when to back-off and up the tree, etc.) can be specified by the user, in reference to the specific datasets and tasks under consideration. For our experiments, the semantics of the tree are as presented in §1.1.

The Chelba method can be thought of as a hierarchical prior in which no smoothing is performed on the tree at all. Only the leaf nodes of the prior’s feature tree are considered, and, if no match can be found between the tuning and prior’s training datasets’ features, a  $\mathcal{N}(0, 1)$  prior is used instead. However, in the new approximate hierarchical model, even if a certain feature in the tuning dataset does not have an analog in the training dataset, we can always back-off until an appropriate match is found, even to the level of the root.

Henceforth, we will use only the approximate hierarchical model in our experiments and discussion.

Table 3: Summary of data used in experiments

Corpus	Genre	Task
UTexas	Bio	Protein
Yapex	Bio	Protein
MUC6	News	Person
MUC7	News	Person
CSPACE	E-mail	Person

### 3 Investigation

#### 3.1 Data, domains and tasks

For our experiments, we have chosen five different corpora (summarized in Table 3). Although each corpus can be considered its own *domain* (due to variations in annotation standards, specific task, date of collection, etc), they can also be roughly grouped into three different *genres*. These are: *abstracts from biological journals* [UT (Bunescu et al., 2004), Yapex (Franzén et al., 2002)]; *news articles* [MUC6 (Fisher et al., 1995), MUC7 (Borthwick et al., 1998)]; and *personal e-mails* [CSPACE (Kraut et al., 2004)]. Each corpus, depending on its *genre*, is labeled with one of two name-finding *tasks*:

- protein names in biological abstracts
- person names in news articles and e-mails

We chose this array of corpora so that we could evaluate our hierarchical prior’s ability to generalize across and incorporate information from a variety of domains, genres and tasks.

In each case, each item (abstract, article or e-mail) was tokenized and each token was hand-labeled as either being part of a name (protein or person) or not, respectively. We used a standard natural language toolkit (Cohen, 2004) to compute tens of thousands of binary features on each of these tokens, encoding such information as capitalization patterns and contextual information from surrounding words. This toolkit produces features of the type described in §1.2 and thus was amenable to our hierarchical prior model. In particular, we chose to use the simplest default, out-of-the-box feature generator and purposefully did not use specifically engineered features, dictionaries, or other techniques commonly employed to boost performance on such tasks. The goal of our experiments was to see to what degree named entity recognition problems naturally conformed to hierarchical methods, and not just to achieve the highest performance possible.

Intra-genre transfer performance evaluated on MUC6

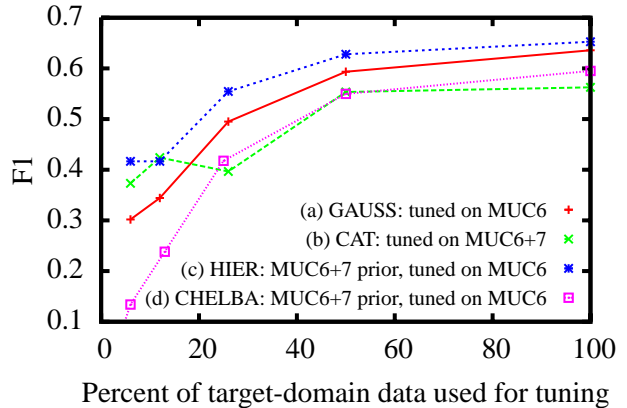


Figure 3: Adding a relevant HIER prior helps compared to the GAUSS baseline ((c) > (a)), while simply CAT’ing or using CHELBA can hurt ((d) ≈ (b) < (a), except with very little data), and never beats HIER ((c) > (b) ≈ (d)).

#### 3.2 Experiments & results

We evaluated the performance of various transfer learning methods on the data and tasks described in §3.1. Specifically, we compared our approximate hierarchical prior model (HIER), implemented as a CRF, against three baselines:

- GAUSS: CRF model tuned on a single domain’s data, using a standard  $\mathcal{N}(0, 1)$  prior
- CAT: CRF model tuned on a concatenation of multiple domains’ data, using a  $\mathcal{N}(0, 1)$  prior
- CHELBA: CRF model tuned on one domain’s data, using a prior trained on a different, related domain’s data (cf. §2.3)

We use token-level *F1* as our main evaluation measure, combining precision and recall into one metric.

##### 3.2.1 Intra-genre, same-task transfer learning

Figure 3 shows the results of an experiment in learning to recognize person names in MUC6 news articles. In this experiment we examined the effect of adding extra data from a different, but related domain from the same genre, namely, MUC7. Line *a* shows the *F1* performance of a CRF model tuned only on the target MUC6 domain (GAUSS) across a range of tuning data sizes. Line *b* shows the same experiment, but this time the CRF model has been tuned on a dataset comprised of a simple concatenation of the training MUC6 data from (*a*), along with a different training set from MUC7 (CAT). We can see that adding extra data in this way, though

Inter-genre transfer performance evaluated on MUC6

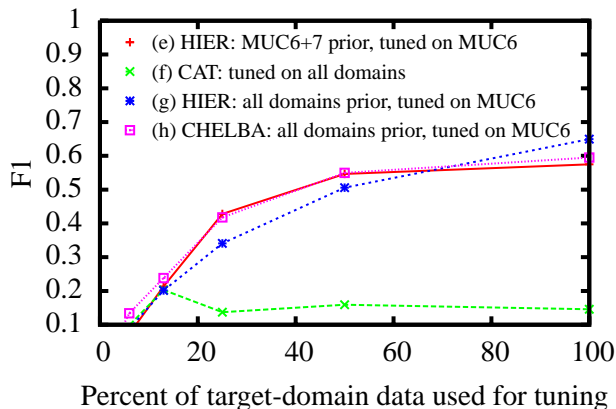


Figure 4: Transfer aware priors CHELBA and HIER effectively filter irrelevant data. Adding more irrelevant data to the priors doesn't hurt ((e)  $\approx$  (g)  $\approx$  (h)), while simply CAT'ing it, in this case, is disastrous ((f)  $\ll$  (e)).

the data is closely related both in domain and task, has actually hurt the performance of our recognizer for training sizes of moderate to large size. This is most likely because, although the MUC6 and MUC7 datasets are closely related, they are still drawn from different distributions and thus cannot be intermingled indiscriminately. Line *c* shows the same combination of MUC6 and MUC7, only this time the datasets have been combined using the HIER prior. In this case, the performance actually does improve, both with respect to the single-dataset trained baseline (*a*) and the naively trained double-dataset (*b*). Finally, line *d* shows the results of the CHELBA prior. Curiously, though the domains are closely related, it does more poorly than even the non-transfer GAUSS. One possible explanation is that, although much of the vocabulary is shared across domains, the interpretation of the features of these words may differ. Since CHELBA doesn't model the hierarchy among features like HIER, it is unable to smooth away these discrepancies. In contrast, we see that our HIER prior is able to successfully combine the relevant parts of data across domains while filtering the irrelevant, and possibly detrimental, ones. This experiment was repeated for other sets of intra-genre tasks, and the results are summarized in §3.2.3.

### 3.2.2 Inter-genre, multi-task transfer learning

In Figure 4 we see that the properties of the hierarchical prior hold even when transferring across

tasks. Here again we are trying to learn to recognize person names in MUC6 e-mails, but this time, instead of adding only other datasets similarly labeled with person names, we are additionally adding biological corpora (UT & YAPEX), labeled not with person names but with protein names instead, along with the CSPACE e-mail and MUC7 news article corpora. The robustness of our prior prevents a model trained on all five domains (*g*) from degrading away from the intra-genre, same-task baseline (*e*), unlike the model trained on concatenated data (*f*). CHELBA (*h*) performs similarly well in this case, perhaps because the domains are so different that almost none of the features match between prior and tuning data, and thus CHELBA backs-off to a standard  $\mathcal{N}(0, 1)$  prior.

This robustness in the face of less similarly related data is very important since these types of transfer methods are most useful when one possesses only very little target domain data. In this situation, it is often difficult to accurately estimate performance and so one would like assurance that any transfer method being applied will not have negative effects.

### 3.2.3 Comparison of HIER prior to baselines

Each scatter plot in Figure 5 shows the relative performance of a baseline method against HIER. Each point represents the results of two experiments: the y-coordinate is the F1 score of the baseline method (shown on the y-axis), while the x-coordinate represents the score of the HIER method in the same experiment. Thus, points lying below the  $y = x$  line represent experiments for which HIER received a higher F1 value than did the baseline. While all three plots show HIER outperforming each of the three baselines, not surprisingly, the non-transfer GAUSS method suffers the worst, followed by the naive concatenation (CAT) baseline. Both methods fail to make any explicit distinction between the source and target domains and thus suffer when the domains differ even slightly from each other. Although the differences are more subtle, the right-most plot of Figure 5 suggests HIER is likewise able to outperform the non-hierarchical CHELBA prior in certain transfer scenarios. CHELBA is able to avoid suffering as much as the other baselines when faced with large difference between domains, but is still unable to capture

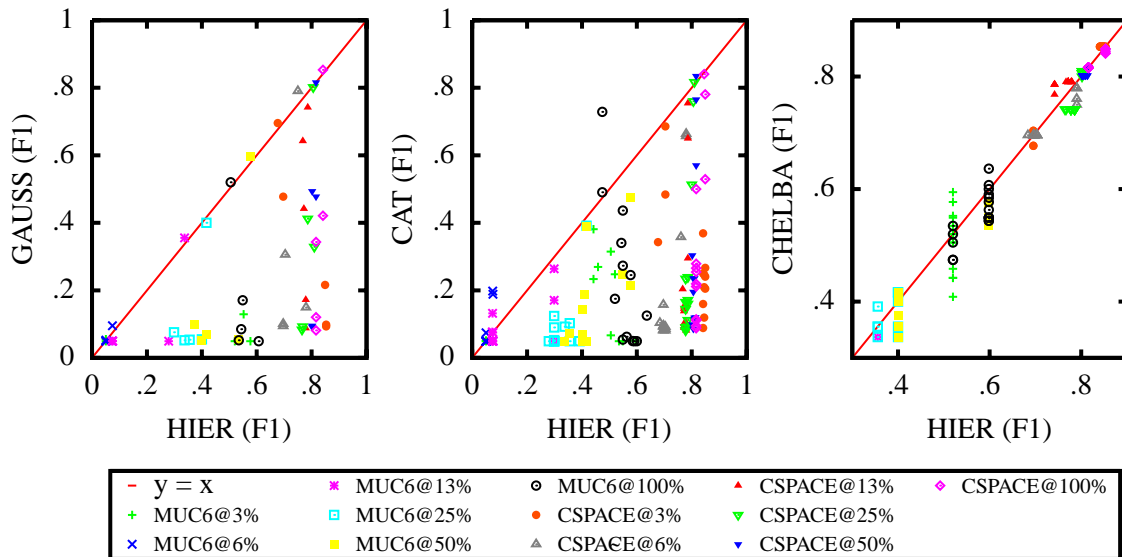


Figure 5: Comparative performance of baseline methods (GAUSS, CAT, CHELBA) vs. HIER prior, as trained on nine prior datasets (both pure and concatenated) of various sample sizes, evaluated on MUC6 and CSPACE datasets. Points below the  $y = x$  line indicate HIER outperforming baselines.

as many dependencies between domains as HIER.

#### 4 Conclusions, related & future work

In this work we have introduced hierarchical feature tree priors for use in transfer learning on named entity extraction tasks. We have provided evidence that motivates these models on intuitive, theoretical and empirical grounds, and have gone on to demonstrate their effectiveness in relation to other, competitive transfer methods. Specifically, we have shown that hierarchical priors allow the user enough flexibility to customize their semantics to a specific problem, while providing enough structure to resist unintended negative effects when used inappropriately. Thus hierarchical priors seem a natural, effective and robust choice for transferring learning across NER datasets and tasks.

Some of the first formulations of the transfer learning problem were presented over 10 years ago (Thrun, 1996; Baxter, 1997). Other techniques have tried to quantify the generalizability of certain features across domains (Daumé III and Marcu, 2006; Jiang and Zhai, 2006), or tried to exploit the common structure of related problems (Ben-David et al., 2007; Schölkopf et al., 2005). Most of this prior work deals with supervised transfer learning, and thus requires labeled source domain data, though there are examples of unsupervised (Arnold

et al., 2007), semi-supervised (Grandvalet and Bengio, 2005; Blitzer et al., 2006), and transductive approaches (Taskar et al., 2003).

Recent work using so-called meta-level priors to transfer information across tasks (Lee et al., 2007), while related, does not take into explicit account the hierarchical structure of these meta-level features often found in NLP tasks. Daumé allows an extra degree of freedom among the features of his domains, implicitly creating a two-level feature hierarchy with one branch for *general* features, and another for *domain specific* ones, but does not extend his hierarchy further (Daumé III, 2007)). Similarly, work on hierarchical penalization (Szafranski et al., 2007) in two-level trees tries to produce models that rely only on a relatively small number of groups of variable, as structured by the tree, as opposed to transferring knowledge between branches themselves.

Our future work is focused on designing an algorithm to optimally choose a smoothing regime for the learned feature trees so as to better exploit the similarities between domains while neutralizing their differences. Along these lines, we are working on methods to reduce the amount of labeled target domain data needed to tune the prior-based models, looking forward to semi-supervised and unsupervised transfer methods.



## References

- Rie K. Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. In *JMLR 6*, pages 1817 – 1853.
- Andrew Arnold, Ramesh Nallapati, and William W. Cohen. 2007. A comparative study of methods for transductive transfer learning. In *Proceedings of the IEEE International Conference on Data Mining (ICDM) 2007 Workshop on Mining and Management of Biological Data*.
- Jonathan Baxter. 1997. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39.
- Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *NIPS 20*, Cambridge, MA. MIT Press.
- John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*, Sydney, Australia.
- A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. 1998. NYU: Description of the MENE named entity system as used in MUC-7.
- R. Bunescu, R. Ge, R. Kate, E. Marcotte, R. Mooney, A. Ramani, and Y. Wong. 2004. Comparative experiments on learning information extractors for proteins and their interactions. In *Journal of AI in Medicine. Data from ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/proteins.tar.gz*.
- Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.
- Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In Dekang Lin and Dekai Wu, editors, *EMNLP 2004*, pages 285–292. ACL.
- S. Chen and R. Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models.
- William W. Cohen. 2004. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. <http://minorthird.sourceforge.net>.
- Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. In *Journal of Artificial Intelligence Research 26*, pages 101–126.
- Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *ACL*.
- David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. 1995. Description of the UMass system as used for MUC-6.
- Kristofer Franzén, Gunnar Eriksson, Fredrik Olsson, Lars Asker, Per Lidn, and Joakim Cöster. 2002. Protein names and how to find them. In *International Journal of Medical Informatics*.
- Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *CAP*, Nice, France.
- Jing Jiang and ChengXiang Zhai. 2006. Exploiting domain structure for named entity recognition. In *Human Language Technology Conference*, pages 74 – 81.
- R. Kraut, S. Fussell, F. Lerch, and J. Espinosa. 2004. Coordination in teams: evidence from a simulated management game.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.
- S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. 2007. Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of International Conference on Machine Learning (ICML)*.
- Einat Minkov, Richard C. Wang, and William W. Cohen. 2005. Extracting personal names from email: Applying named entity recognition to informal text. In *HLT/EMNLP*.
- Rajat Raina, Andrew Y. Ng, and Daphne Koller. 2006. Transfer learning by constructing informative priors. In *ICML 22*.
- Bernhard Schölkopf, Florian Steinke, and Volker Blanz. 2005. Object correspondence as a machine learning problem. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 776–783, New York, NY, USA. ACM.
- Charles Sutton and Andrew McCallum. 2005. Composition of conditional random fields for transfer learning. In *HLT/EMNLP*.
- M. Szafranski, Y. Grandvalet, and P. Morizet-Mahoudeaux. 2007. Hierarchical penalization. In *Advances in Neural Information Processing Systems 20*. MIT press.
- B. Taskar, M.-F. Wong, and D. Koller. 2003. Learning on the test data: Leveraging ‘unseen’ features. In *Proc. Twentieth International Conference on Machine Learning (ICML)*.
- Sebastian Thrun. 1996. Is learning the  $n$ -th thing any easier than learning the first? In *NIPS*, volume 8, pages 640–646. MIT.
- J. Zhang, Z. Ghahramani, and Y. Yang. 2005. Learning multiple related tasks using latent independent component analysis.

# Refining Event Extraction through Cross-document Inference

Heng Ji

Computer Science Department  
New York University  
New York, NY 10003, USA  
(hengji, grishman)@cs.nyu.edu

Ralph Grishman

## Abstract

We apply the hypothesis of “One Sense Per Discourse” (Yarowsky, 1995) to information extraction (IE), and extend the scope of “discourse” from one single document to a cluster of topically-related documents. We employ a similar approach to propagate consistent event arguments across sentences and documents. Combining global evidence from related documents with local decisions, we design a simple scheme to conduct cross-document inference for improving the ACE event extraction task<sup>1</sup>. Without using any additional labeled data this new approach obtained 7.6% higher F-Measure in trigger labeling and 6% higher F-Measure in argument labeling over a state-of-the-art IE system which extracts events independently for each sentence.

## 1 Introduction

Identifying events of a particular type within individual documents – ‘classical’ information extraction – remains a difficult task. Recognizing the different forms in which an event may be expressed, distinguishing events of different types, and finding the arguments of an event are all challenging tasks.

Fortunately, many of these events will be reported multiple times, in different forms, both within the same document and within *topically-related* documents (i.e. a collection of documents sharing participants in potential events). We can

take advantage of these alternate descriptions to improve event extraction in the original document, by favoring consistency of interpretation across sentences and documents. Several recent studies involving specific event types have stressed the benefits of going beyond traditional single-document extraction; in particular, Yangarber (2006) has emphasized this potential in his work on medical information extraction. In this paper we demonstrate that appreciable improvements are possible over the variety of event types in the ACE (Automatic Content Extraction) evaluation through the use of cross-sentence and cross-document evidence.

As we shall describe below, we can make use of consistency at several levels: consistency of word sense across different instances of the same word in related documents, and consistency of arguments and roles across different mentions of the same or related events. Such methods allow us to build dynamic background knowledge as required to interpret a document and can compensate for the limited annotated training data which can be provided for each event type.

## 2 Task and Baseline System

### 2.1 ACE Event Extraction Task

The event extraction task we are addressing is that of the Automatic Content Extraction (ACE) evaluations<sup>2</sup>. ACE defines the following terminology:

---

<sup>1</sup> <http://www.nist.gov/speech/tests/ace/>

---

<sup>2</sup> In this paper we don’t consider event mention coreference resolution and so don’t distinguish event mentions and events.

**entity**: an object or a set of objects in one of the semantic categories of interest

**mention**: a reference to an entity (typically, a noun phrase)

**event trigger**: the main word which most clearly expresses an event occurrence

**event arguments**: the mentions that are involved in an event (participants)

**event mention**: a phrase or sentence within which an event is described, including trigger and arguments

The 2005 ACE evaluation had 8 types of events, with 33 subtypes; for the purpose of this paper, we will treat these simply as 33 distinct event types. For example, for a sentence:

*Barry Diller on Wednesday quit as chief of Vivendi Universal Entertainment.*

the event extractor should detect a “Personnel\_End-Position” event mention, with the trigger word, the position, the person who quit the position, the organization, and the time during which the event happened:

	Trigger	<i>Quit</i>
Arguments	Role = Person	<i>Barry Diller</i>
	Role = Organization	<i>Vivendi Universal Entertainment</i>
	Role = Position	<i>Chief</i>
	Role = Time-within	<i>Wednesday</i>

Table 1. Event Extraction Example

We define the following standards to determine the *correctness* of an event mention:

- *A trigger is correctly labeled* if its event type and offsets match a reference trigger.
- *An argument is correctly identified* if its event type and offsets match any of the reference argument mentions.
- *An argument is correctly identified and classified* if its event type, offsets, and role match any of the reference argument mentions.

## 2.2 A Baseline Within-Sentence Event Tagger

We use a state-of-the-art English IE system as our baseline (Grishman et al., 2005). This system extracts events independently for each sentence. Its training and test procedures are as follows.

The system combines pattern matching with statistical models. For every event mention in the ACE training corpus, patterns are constructed based on the sequences of constituent heads separating the trigger and arguments. In addition, a set of Maximum Entropy based classifiers are trained:

- **Trigger Labeling**: to distinguish event mentions from non-event-mentions, to classify event mentions by type;
- **Argument Classifier**: to distinguish arguments from non-arguments;
- **Role Classifier**: to classify arguments by argument role.
- **Reportable-Event Classifier**: Given a trigger, an event type, and a set of arguments, to determine whether there is a reportable event mention.

In the test procedure, each document is scanned for instances of triggers from the training corpus. When an instance is found, the system tries to match the environment of the trigger against the set of patterns associated with that trigger. This pattern-matching process, if successful, will assign some of the mentions in the sentence as arguments of a potential event mention. The argument classifier is applied to the remaining mentions in the sentence; for any argument passing that classifier, the role classifier is used to assign a role to it. Finally, once all arguments have been assigned, the reportable-event classifier is applied to the potential event mention; if the result is successful, this event mention is reported.

## 3 Motivations

In this section we shall present our motivations based on error analysis for the baseline event tagger.

### 3.1 One Trigger Sense Per Cluster

Across a heterogeneous document corpus, a particular verb can sometimes be trigger and sometimes not, and can represent different event types. However, for a collection of topically-related documents, the distribution may be much more convergent. We investigate this hypothesis by automatically obtaining 25 related documents for each test text. The statistics of some trigger examples are presented in table 2.

Candidate Triggers		Event Type	Perc./Freq. as trigger in ACE training corpora	Perc./Freq. as trigger in test document	Perc./Freq. as trigger in test + related documents
Correct Event Triggers	<i>advance</i>	Movement_Transport	31% of 16	50% of 2	88.9% of 27
	<i>fire</i>	Personnel_End-Position	7% of 81	100% of 2	100% of 10
	<i>fire</i>	Conflict_Attack	54% of 81	100% of 3	100% of 19
	<i>replace</i>	Personnel_End-Position	5% of 20	100% of 1	83.3% of 6
	<i>form</i>	Business_Start-Org	12% of 8	100% of 2	100% of 23
	<i>talk</i>	Contact_Meet	59% of 74	100% of 4	100% of 26
Incorrect Event Triggers	<i>hurt</i>	Life_Injure	24% of 33	0% of 2	0% of 7
	<i>execution</i>	Life_Die	12% of 8	0% of 4	4% of 24

Table 2. Examples: Percentage of a Word as Event Trigger in Different Data Collections

As we can see from the table, the likelihood of a candidate word being an event trigger in the test document is closer to its distribution in the collection of related documents than the uniform training corpora. So if we can determine the sense (event type) of a word in the related documents, this will allow us to infer its sense in the test document. In this way related documents can help recover event mentions missed by within-sentence extraction.

For example, in a document about “the advance into Baghdad”:

**Example 1:**

**[Test Sentence]**

*Most US army commanders believe it is critical to pause the breakneck **advance** towards Baghdad to secure the supply lines and make sure weapons are operable and troops resupplied....*

**[Sentences from Related Documents]**

*British and US forces report gains in the **advance** on Baghdad and take control of Umm Qasr, despite a fierce sandstorm which slows another flank.*

...

The baseline event tagger is not able to detect “advance” as a “Movement\_Transport” event trigger because there is no pattern “advance towards [Place]” in the ACE training corpora (“advance” by itself is too ambiguous). The training data, however, does include the pattern “advance on [Place]”, which allows the instance of “advance” in the related documents to be successfully identified with high confidence by pattern matching as an event. This provides us much stronger “feedback” confidence in tagging ‘advance’ in the test sentence as a correct trigger.

On the other hand, if a word is not tagged as an event trigger in most related documents, then it’s less likely to be correct in the test sentence despite its high local confidence. For example, in a document about “assessment of Russian president Putin”:

**Example 2:**

**[Test Sentence]**

*But few at the Kremlin forum suggested that Putin’s own standing among voters will be **hurt** by Russia’s apparent diplomacy failures.*

**[Sentences from Related Documents]**

*Putin boosted ties with the United States by throwing his support behind its war on terrorism after the Sept. 11 attacks, but the Iraq war has **hurt** the relationship.*

...

The word “hurt” in the test sentence is mistakenly identified as a “Life\_Injure” trigger with high local confidence (because the within-sentence extractor misanalyzes “voters” as the object of “hurt” and so matches the pattern “[Person] be hurt”). Based on the fact that many other instances of “hurt” are not “Life\_Injure” triggers in the related documents, we can successfully remove this wrong event mention in the test document.

### 3.2 One Argument Role Per Cluster

Inspired by the observation about trigger distribution, we propose a similar hypothesis – one argument role per cluster for event arguments. In other words, each entity plays the same argument role, or no role, for events with the same type in a collection of related documents. For example,

### Example 3:

#### [Test Sentence]

*Vivendi* earlier this week confirmed months of press speculation that it planned to **shed** its entertainment assets by the end of the year.

#### [Sentences from Related Documents]

*Vivendi* has been trying to **sell** assets to pay off huge debt, estimated at the end of last month at more than \$13 billion.

Under the reported plans, Blackstone Group would **buy** *Vivendi*'s theme park division, including Universal Studios Hollywood, Universal Orlando in Florida...

...

The above test sentence doesn't include an explicit trigger word to indicate "Vivendi" as a "seller" of a "Transaction\_Transfer-Ownership" event mention, but "Vivendi" is correctly identified as "seller" in many other related sentences (by matching patterns "[Seller] sell" and "buy [Seller]'s"). So we can incorporate such additional information to enhance the confidence of "Vivendi" as a "seller" in the test sentence.

On the other hand, we can remove spurious arguments with low cross-document frequency and confidence. In the following example,

### Example 4:

#### [Test Sentence]

*The Davao Medical Center*, a regional government hospital, recorded 19 deaths with 50 wounded.

"the Davao Medical Center" is mistakenly tagged as "Place" for a "Life\_Die" event mention. But the same annotation for this mention doesn't appear again in the related documents, so we can determine it's a spurious argument.

## 4 System Approach Overview

Based on the above motivations we propose to incorporate global evidence from a cluster of related documents to refine local decisions. This section gives more details about the baseline within-sentence event tagger, and the information retrieval system we use to obtain related documents. In the next section we shall focus on describing the inference procedure.

### 4.1 System Pipeline

Figure 1 depicts the general procedure of our approach.  $EMSet$  represents a set of event mentions which is gradually updated.

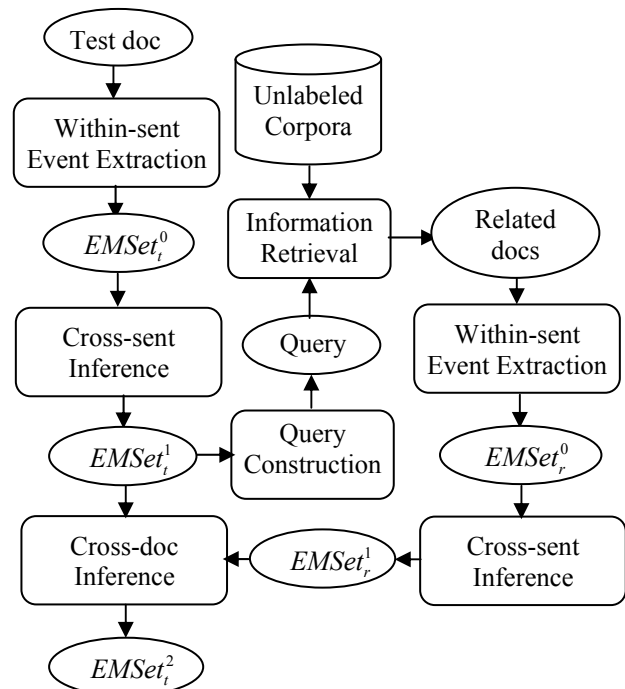


Figure 1. Cross-doc Inference for Event Extraction

### 4.2 Within-Sentence Event Extraction

For each event mention in a test document  $t$ , the baseline Maximum Entropy based classifiers produce three types of confidence values:

- $LConf(trigger, etype)$ : The probability of a string  $trigger$  indicating an event mention with type  $etype$ ; if the event mention is produced by pattern matching then assign confidence 1.
- $LConf(arg, etype)$ : The probability that a mention  $arg$  is an argument of some particular event type  $etype$ .
- $LConf(arg, etype, role)$ : If  $arg$  is an argument with event type  $etype$ , the probability of  $arg$  having some particular  $role$ .

We apply within-sentence event extraction to get an initial set of event mentions  $EMSet_t^0$ , and conduct cross-sentence inference (details will be presented in section 5) to get an updated set of event mentions  $EMSet_t^1$ .

### 4.3 Information Retrieval

We then use the INDRI retrieval system (Strohman et al., 2005) to obtain the top N (N=25 in this pa-

per<sup>3</sup>) related documents. We construct an INDRI query from the triggers and arguments, each weighted by local confidence and frequency in the test document. For each argument we also add other names coreferential with or bearing some ACE relation to the argument.

For each related document  $r$  returned by INDRI, we repeat the within-sentence event extraction and cross-sentence inference procedure, and get an expanded event mention set  $EMSet_{t+r}^1$ . Then we apply cross-document inference to  $EMSet_{t+r}^1$  and get the final event mention output  $EMSet_t^2$ .

## 5 Global Inference

The central idea of inference is to obtain document-wide and cluster-wide statistics about the frequency with which triggers and arguments are associated with particular types of events, and then use this information to correct event and argument identification and classification.

For a set of event mentions we tabulate the following document-wide and cluster-wide confidence-weighted frequencies:

- for each trigger string, the frequency with which it appears as the trigger of an event of a particular type;
- for each event argument string and the names coreferential with or related to the argument, the frequency of the event type;
- for each event argument string and the names coreferential with or related to the argument, the frequency of the event type and role.

Besides these frequencies, we also define the following *margin* metric to compute the confidence of the best (most frequent) event type or role:

$$\text{Margin} = \frac{(\text{WeightedFrequency}(\text{most frequent value}) - \text{WeightedFrequency}(\text{second most freq value}))}{\text{WeightedFrequency}(\text{second most freq value})}$$

A large margin indicates greater confidence in the most frequent value. We summarize the frequency and confidence metrics in Table 3.

Based on these confidence metrics, we designed the inference rules in Table 4. These rules are applied in the order (1) to (9) based on the principle of improving ‘local’ information before global

propagation. Although the rules may seem complex, they basically serve two functions:

- to remove triggers and arguments with low (local or cluster-wide) confidence;
- to adjust trigger and argument identification and classification to achieve (document-wide or cluster-wide) consistency.

## 6 Experimental Results and Analysis

In this section we present the results of applying this inference method to improve ACE event extraction.

### 6.1 Data

We used 10 newswire texts from ACE 2005 training corpora (from March to May of 2003) as our development set, and then conduct blind test on a separate set of 40 ACE 2005 newswire texts. For each test text we retrieved 25 related texts from English TDT5 corpus which in total consists of 278,108 texts (from April to September of 2003).

### 6.2 Confidence Metric Thresholding

We select the thresholds ( $\delta_k$  with  $k=1\sim 13$ ) for various confidence metrics by optimizing the F-measure score of each rule on the development set, as shown in Figure 2 and 3 as follows.

Each curve in Figure 2 and 3 shows the effect on precision and recall of varying the threshold for an individual rule.

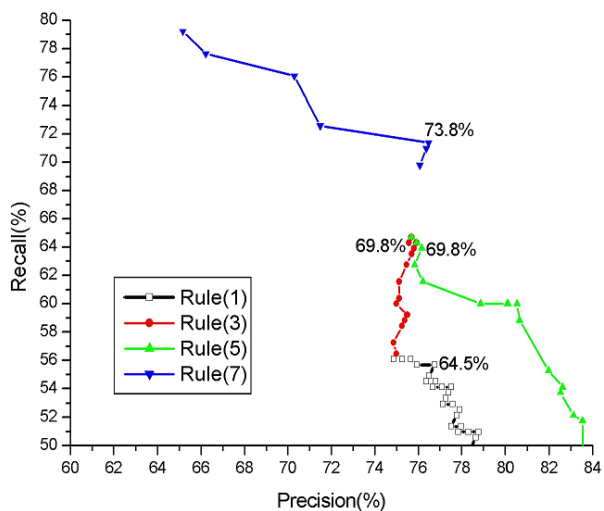


Figure 2. Trigger Labeling Performance with Confidence Thresholding on Dev Set

<sup>3</sup> We tested different  $N \in [10, 75]$  on dev set; and  $N=25$  achieved best gains.

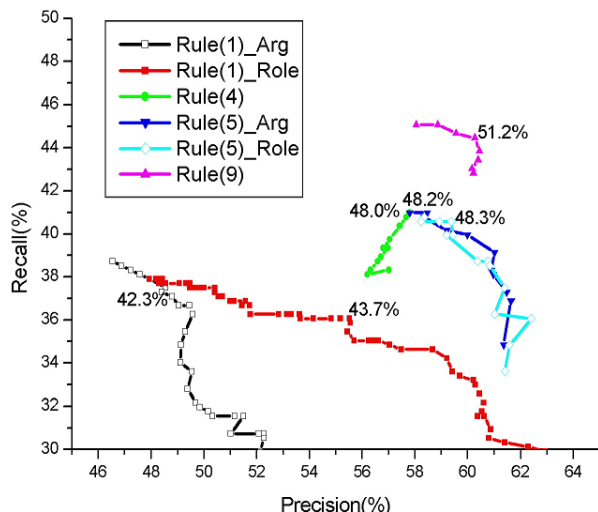


Figure 3. Argument Labeling Performance with Confidence Thresholding on Dev Set

The labeled point on each curve shows the best F-measure that can be obtained on the development set by adjusting the threshold for that rule. The gain obtained by applying successive rules can be seen in the progression of successive points towards higher recall and, for argument labeling, precision<sup>4</sup>.

### 6.3 Overall Performance

Table 5 shows the overall Precision (P), Recall (R) and F-Measure (F) scores for the blind test set. In addition, we also measured the performance of two human annotators who prepared the ACE 2005 training data on 28 newswire texts (a subset of the blind test set). The final key was produced by review and adjudication of the two annotations.

Both cross-sentence and cross-document inferences provided significant improvement over the baseline with local confidence thresholds controlled.

We conducted the Wilcoxon Matched-Pairs Signed-Ranks Test on a document basis. The results show that the improvement using cross-sentence inference is significant at a 99.9% confidence level for both trigger and argument labeling; adding cross-document inference is significant at a 99.9% confidence level for trigger labeling and 93.4% confidence level for argument labeling.

<sup>4</sup> We didn't show the classification adjusting rules (2), (6) and (8) here because of their relatively small impact on dev set.

### 6.4 Discussion

From table 5 we can see that for trigger labeling our approach dramatically enhanced recall (22.9% improvement) with some loss (7.4%) in precision. This precision loss was much larger than that for the development set (0.3%). This indicates that the trigger propagation thresholds optimized on the development set were too low for the blind test set and thus more spurious triggers got propagated. The improved trigger labeling is better than one human annotator and only 4.7% worse than another.

For argument labeling we can see that cross-sentence inference improved both identification (3.7% higher F-Measure) and classification (6.1% higher accuracy); and cross-document inference mainly provided further gains (1.9%) in classification. This shows that identification consistency may be achieved within a narrower context while the classification task favors more global background knowledge in order to solve some difficult cases. This matches the situation of human annotation as well: we may decide whether a mention is involved in some particular event or not by reading and analyzing the target sentence itself; but in order to decide the argument's role we may need to frequently refer to wider discourse in order to infer and confirm our decision. In fact sometimes it requires us to check more similar web pages or even wikipedia databases. This was exactly the intuition of our approach. We should also note that human annotators label arguments based on perfect entity mentions, but our system used the output from the IE system. So the gap was also partially due to worse entity detection.

Error analysis on the inference procedure shows that the propagation rules (3), (4), (7) and (9) produced a few extra false alarms. For trigger labeling, most of these errors appear for support verbs such as "take" and "get" which can only represent an event mention together with other verbs or nouns. Some other errors happen on nouns and adjectives. These are difficult tasks even for human annotators. As shown in table 5 the inter-annotator agreement on trigger identification is only about 40%. Besides some obvious overlooked cases (it's probably difficult for a human to remember 33 different event types during annotation), most difficulties were caused by judging generic verbs, nouns and adjectives.

Performance System/Human	Trigger Identification +Classification			Argument Identification			Argument Classification Accuracy	Argument Identification +Classification		
	P	R	F	P	R	F		P	R	F
Within-Sentence IE with Rule (1) (Baseline)	67.6	53.5	59.7	47.8	38.3	42.5	86.0	41.2	32.9	36.6
Cross-sentence Inference	64.3	59.4	61.8	54.6	38.5	45.1	90.2	49.2	34.7	40.7
Cross-sentence+ Cross-doc Inference	<b>60.2</b>	<b>76.4</b>	<b>67.3</b>	<b>55.7</b>	<b>39.5</b>	<b>46.2</b>	<b>92.1</b>	<b>51.3</b>	<b>36.4</b>	<b>42.6</b>
Human Annotator1	59.2	59.4	59.3	60.0	69.4	64.4	85.8	51.6	59.5	55.3
Human Annotator2	69.2	75.0	72.0	62.7	85.4	72.3	86.3	54.1	73.7	62.4
Inter-Annotator Agreement	41.9	38.8	40.3	55.2	46.7	50.6	91.7	50.6	42.9	46.4

Table 5. Overall Performance on Blind Test Set (%)

In fact, compared to a statistical tagger trained on the corpus after expert adjudication, a human annotator tends to make more mistakes in trigger classification. For example it’s hard to decide whether “named” represents a “Personnel\_Nominate” or “Personnel\_Start-Position” event mention; “hacked to death” represents a “Life\_Die” or “Conflict\_Attack” event mention without following more specific annotation guidelines.

## 7 Related Work

The trigger labeling task described in this paper is in part a task of word sense disambiguation (WSD), so we have used the idea of sense consistency introduced in (Yarowsky, 1995), extending it to operate across related documents.

Almost all the current event extraction systems focus on processing single documents and, except for coreference resolution, operate a sentence at a time (Grishman et al., 2005; Ahn, 2006; Hardy et al., 2006).

We share the view of using global inference to improve event extraction with some recent research. Yangarber et al. (Yangarber and Jokipii, 2005; Yangarber, 2006; Yangarber et al., 2007) applied cross-document inference to correct local extraction results for disease name, location and start/end time. Mann (2007) encoded specific inference rules to improve extraction of CEO (name, start year, end year) in the MUC management succession task. In addition, Patwardhan and Riloff (2007) also demonstrated that selectively applying event patterns to relevant regions can improve MUC event extraction. We expand the idea to more general event types and use informa-

tion retrieval techniques to obtain wider background knowledge from related documents.

## 8 Conclusion and Future Work

One of the initial goals for IE was to create a database of relations and events from the entire input corpus, and allow further logical reasoning on the database. The artificial constraint that extraction should be done independently for each document was introduced in part to simplify the task and its evaluation. In this paper we propose a new approach to break down the document boundaries for event extraction. We gather together event extraction results from a set of related documents, and then apply inference and constraints to enhance IE performance.

In the short term, the approach provides a platform for many byproducts. For example, we can naturally get an event-driven summary for the collection of related documents; the sentences including high-confidence events can be used as additional training data to bootstrap the event tagger; from related events in different timeframes we can derive entailment rules; the refined consistent events can serve better for other NLP tasks such as template based question-answering. The aggregation approach described here can be easily extended to improve relation detection and coreference resolution (two argument mentions referring to the same role of related events are likely to corefer). Ultimately we would like to extend the system to perform essential, although probably lightweight, event prediction.



$XSent-Trigger-Freq(trigger, etype)$	The weighted frequency of string $trigger$ appearing as the trigger of an event of type $etype$ across all sentences within a document
$XDoc-Trigger-Freq(trigger, etype)$	The weighted frequency of string $trigger$ appearing as the trigger of an event of type $etype$ across all documents in a cluster
$XDoc-Trigger-BestFreq(trigger)$	Maximum over all $etypes$ of $XDoc-Trigger-Freq(trigger, etype)$
$XDoc-Arg-Freq(arg, etype)$	The weighted frequency of $arg$ appearing as an argument of an event of type $etype$ across all documents in a cluster
$XDoc-Role-Freq(arg, etype, role)$	The weighted frequency of $arg$ appearing as an argument of an event of type $etype$ with role $role$ across all documents in a cluster
$XDoc-Role-BestFreq(arg)$	Maximum over all $etypes$ and roles of $XDoc-Role-Freq(arg, etype, role)$
$XSent-Trigger-Margin(trigger)$	The margin value of $trigger$ in $XSent-Trigger-Freq$
$XDoc-Trigger-Margin(trigger)$	The margin value of $trigger$ in $XDoc-Trigger-Freq$
$XDoc-Role-Margin(arg)$	The margin value of $arg$ in $XDoc-Role-Freq$

Table 3. Global Frequency and Confidence Metrics

<b>Rule (1): Remove Triggers and Arguments with Low Local Confidence</b>
If $LConf(trigger, etype) < \delta_1$ , then delete the whole event mention $EM$ ; If $LConf(arg, etype) < \delta_2$ or $LConf(arg, etype, role) < \delta_3$ , then delete $arg$ .
<b>Rule (2): Adjust Trigger Classification to Achieve Document-wide Consistency</b>
If $XSent-Trigger-Margin(trigger) > \delta_4$ , then propagate the most frequent $etype$ to all event mentions with $trigger$ in the document; and correct roles for corresponding arguments.
<b>Rule (3): Adjust Trigger Identification to Achieve Document-wide Consistency</b>
If $LConf(trigger, etype) > \delta_5$ , then propagate $etype$ to all unlabeled strings $trigger$ in the document.
<b>Rule (4): Adjust Argument Identification to Achieve Document-wide Consistency</b>
If $LConf(arg, etype) > \delta_6$ , then in the document, for each sentence containing an event mention $EM$ with $etype$ , add any unlabeled mention in that sentence with the same head as $arg$ as an argument of $EM$ with $role$ .
<b>Rule (5): Remove Triggers and Arguments with Low Cluster-wide Confidence</b>
If $XDoc-Trigger-Freq(trigger, etype) < \delta_7$ , then delete $EM$ ; If $XDoc-Arg-Freq(arg, etype) < \delta_8$ or $XDoc-Role-Freq(arg, etype, role) < \delta_9$ , then delete $arg$ .
<b>Rule (6): Adjust Trigger Classification to Achieve Cluster-wide Consistency</b>
If $XDoc-Trigger-Margin(trigger) > \delta_{10}$ , then propagate most frequent $etype$ to all event mentions with $trigger$ in the cluster; and correct roles for corresponding arguments.
<b>Rule (7): Adjust Trigger Identification to Achieve Cluster-wide Consistency</b>
If $XDoc-Trigger-BestFreq(trigger) > \delta_{11}$ , then propagate $etype$ to all unlabeled strings $trigger$ in the cluster, override the results of Rule (3) if conflict.
<b>Rule (8): Adjust Argument Classification to Achieve Cluster-wide Consistency</b>
If $XDoc-Role-Margin(arg) > \delta_{12}$ , then propagate the most frequent $etype$ and $role$ to all arguments with the same head as $arg$ in the entire cluster.
<b>Rule (9): Adjust Argument Identification to Achieve Cluster-wide Consistency</b>
If $XDoc-Role-BestFreq(arg) > \delta_{13}$ , then in the cluster, for each sentence containing an event mention $EM$ with $etype$ , add any unlabeled mention in that sentence with the same head as $arg$ as an argument of $EM$ with $role$ .

Table 4. Probabilistic Inference Rule

## Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency under Contract No. HR0011-06-C-0023, and the Na-

tional Science Foundation under Grant IIS-00325657. Any opinions, findings and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the U. S. Government.

## References

- David Ahn. 2006. The stages of event extraction. *Proc. COLING/ACL 2006 Workshop on Annotating and Reasoning about Time and Events*. Sydney, Australia.
- Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYU's English ACE 2005 System Description. *Proc. ACE 2005 Evaluation Workshop*. Washington, US.
- Hilda Hardy, Vika Kanchakouskaya and Tomek Strzalowski. 2006. Automatic Event Classification Using Surface Text Features. *Proc. AAAI06 Workshop on Event Extraction and Synthesis*. Boston, Massachusetts. US.
- Gideon Mann. 2007. Multi-document Relationship Fusion via Constraints on Probabilistic Databases. *Proc. HLT/NAACL 2007*. Rochester, NY, US.
- Siddharth Patwardhan and Ellen Riloff. 2007. Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions. *Proc. EMNLP 2007*. Prague, Czech Republic.
- Trevor Strohman, Donald Metzler, Howard Turtle and W. Bruce Croft. 2005. Indri: A Language-model based Search Engine for Complex Queries (extended version). *Technical Report IR-407, CIIR*, Umass Amherst, US.
- Roman Yangarber, Clive Best, Peter von Etter, Flavio Fuart, David Horby and Ralf Steinberger. 2007. Combining Information about Epidemic Threats from Multiple Sources. *Proc. RANLP 2007 workshop on Multi-source, Multilingual Information Extraction and Summarization*. Borovets, Bulgaria.
- Roman Yangarber. 2006. Verification of Facts across Document Boundaries. *Proc. International Workshop on Intelligent Information Access*. Helsinki, Finland.
- Roman Yangarber and Lauri Jokipii. 2005. Redundancy-based Correction of Automatically Extracted Facts. *Proc. HLT/EMNLP 2005*. Vancouver, Canada.
- David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. *Proc. ACL 1995*. Cambridge, MA, US.

# Learning Document-Level Semantic Properties from Free-text Annotations

S.R.K. Branavan Harr Chen Jacob Eisenstein Regina Barzilay

Computer Science and Artificial Intelligence Laboratory

Massachusetts Institute of Technology

{branavan, harr, jacob, regina}@csail.mit.edu

## Abstract

This paper demonstrates a new method for leveraging free-text annotations to infer semantic properties of documents. Free-text annotations are becoming increasingly abundant, due to the recent dramatic growth in semi-structured, user-generated online content. An example of such content is product reviews, which are often annotated by their authors with pros/cons keyphrases such as “a real bargain” or “good value.” To exploit such noisy annotations, we simultaneously find a hidden paraphrase structure of the keyphrases, a model of the document texts, and the underlying semantic properties that link the two. This allows us to predict properties of unannotated documents. Our approach is implemented as a hierarchical Bayesian model with joint inference, which increases the robustness of the keyphrase clustering and encourages the document model to correlate with semantically meaningful properties. We perform several evaluations of our model, and find that it substantially outperforms alternative approaches.

## 1 Introduction

A central problem in language understanding is transforming raw text into structured representations. Learning-based approaches have dramatically increased the scope and robustness of this type of automatic language processing, but they are typically dependent on large expert-annotated datasets, which are costly to produce. In this paper, we show how novice-generated free-text annotations available online can be leveraged to automatically infer document-level semantic properties.

With the rapid increase of online content created by end users, noisy free-text annotations have

pros/cons: <i>great nutritional value</i> ... combines it all: an amazing product, quick and friendly service, cleanliness, great nutrition ...
pros/cons: <i>a bit pricey, healthy</i> ... is an awesome place to go if you are health conscious. They have some really great low calorie dishes and they publish the calories and fat grams per serving.

Figure 1: Excerpts from online restaurant reviews with pros/cons phrase lists. Both reviews discuss healthiness, but use different keyphrases.

become widely available (Vickery and Wunsch-Vincent, 2007; Sterling, 2005). For example, consider reviews of consumer products and services. Often, such reviews are annotated with *keyphrase* lists of pros and cons. We would like to use these keyphrase lists as training labels, so that the properties of unannotated reviews can be predicted. Having such a system would facilitate structured access and summarization of this data. However, novice-generated keyphrase annotations are incomplete descriptions of their corresponding review texts. Furthermore, they lack consistency: the same underlying property may be expressed in many ways, e.g., “healthy” and “great nutritional value” (see Figure 1). To take advantage of such noisy labels, a system must both uncover their hidden clustering into *properties*, and learn to predict these properties from review text.

This paper presents a model that addresses both problems simultaneously. We assume that both the document text and the selection of keyphrases are governed by the underlying hidden properties of the document. Each property indexes a language model, thus allowing documents that incorporate the same

property to share similar features. In addition, each keyphrase is associated with a property; keyphrases that are associated with the same property should have similar distributional and surface features.

We link these two ideas in a joint hierarchical Bayesian model. Keyphrases are clustered based on their distributional and lexical properties, and a hidden topic model is applied to the document text. Crucially, the keyphrase clusters and document topics are linked, and inference is performed jointly. This increases the robustness of the keyphrase clustering, and ensures that the inferred hidden topics are indicative of salient semantic properties.

Our model is broadly applicable to many scenarios where documents are annotated in a noisy manner. In this work, we apply our method to a collection of reviews in two categories: restaurants and cell phones. The training data consists of review text and the associated pros/cons lists. We then evaluate the ability of our model to predict review properties when the pros/cons list is hidden. Across a variety of evaluation scenarios, our algorithm consistently outperforms alternative strategies by a wide margin.

## 2 Related Work

**Review Analysis** Our approach relates to previous work on property extraction from reviews (Popescu et al., 2005; Hu and Liu, 2004; Kim and Hovy, 2006). These methods extract lists of phrases, which are analogous to the keyphrases we use as input to our algorithm. However, our approach is distinguished in two ways: first, we are able to predict keyphrases beyond those that appear verbatim in the text. Second, our approach learns the relationships between keyphrases, allowing us to draw direct comparisons between reviews.

**Bayesian Topic Modeling** One aspect of our model views properties as distributions over words in the document. This approach is inspired by methods in the topic modeling literature, such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003), where topics are treated as hidden variables that govern the distribution of words in a text. Our algorithm extends this notion by biasing the induced hidden topics toward a clustering of known keyphrases. Tying these two information sources together enhances the robustness of the hidden topics, thereby increasing

the chance that the induced structure corresponds to semantically meaningful properties.

Recent work has examined coupling topic models with explicit supervision (Blei and McAuliffe, 2007; Titov and McDonald, 2008). However, such approaches assume that the documents are labeled within a predefined annotation structure, *e.g.*, the properties of food, ambiance, and service for restaurants. In contrast, we address free-text annotations created by end users, without known semantic properties. Rather than requiring a predefined annotation structure, our model infers one from the data.

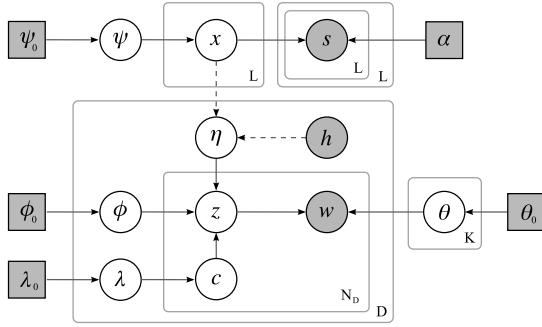
## 3 Problem Formulation

We formulate our problem as follows. We assume a dataset composed of documents with associated keyphrases. Each document may be marked with multiple keyphrases that express unseen semantic properties. Across the entire collection, several keyphrases may express the same property. The keyphrases are also incomplete — review texts often express properties that are not mentioned in their keyphrases. At training time, our model has access to both text and keyphrases; at test time, the goal is to predict the properties supported by a previously unseen document. We can then use this property list to generate an appropriate set of keyphrases.

## 4 Model Description

Our approach leverages both keyphrase clustering and distributional analysis of the text in a joint, hierarchical Bayesian model. Keyphrases are drawn from a set of clusters; words in the documents are drawn from language models indexed by a set of topics, where the topics correspond to the keyphrase clusters. Crucially, we bias the assignment of hidden topics in the text to be similar to the topics represented by the keyphrases of the document, but we permit some words to be drawn from other topics not represented by the keyphrases. This flexibility in the coupling allows the model to learn effectively in the presence of incomplete keyphrase annotations, while still encouraging the keyphrase clustering to cohere with the topics supported by the text.

We train the model on documents annotated with keyphrases. During training, we learn a hidden topic model from the text; each topic is also asso-



- $\psi$  – keyphrase cluster model
- $x$  – keyphrase cluster assignment
- $s$  – keyphrase similarity values
- $h$  – document keyphrases
- $\eta$  – document keyphrase topics
- $\lambda$  – probability of selecting  $\eta$  instead of  $\phi$
- $c$  – selects between  $\eta$  and  $\phi$  for word topics
- $\phi$  – document topic model
- $z$  – word topic assignment
- $\theta$  – language models of each topic
- $w$  – document words

$$\begin{aligned} \psi &\sim \text{Dirichlet}(\psi_0) \\ x_\ell &\sim \text{Multinomial}(\psi) \\ s_{\ell, \ell'} &\sim \begin{cases} \text{Beta}(\alpha_{=}) & \text{if } x_\ell = x_{\ell'} \\ \text{Beta}(\alpha_{\neq}) & \text{otherwise} \end{cases} \\ \eta_d &= [\eta_{d,1} \dots \eta_{d,K}]^T \\ &\text{where} \\ \eta_{d,k} &\propto \begin{cases} 1 & \text{if } x_\ell = k \text{ for any } \ell \in h_d \\ 0 & \text{otherwise} \end{cases} \\ \lambda &\sim \text{Beta}(\lambda_0) \\ c_{d,n} &\sim \text{Bernoulli}(\lambda) \\ \phi_d &\sim \text{Dirichlet}(\phi_0) \\ z_{d,n} &\sim \begin{cases} \text{Multinomial}(\eta_d) & \text{if } c_{d,n} = 1 \\ \text{Multinomial}(\phi_d) & \text{otherwise} \end{cases} \\ \theta_k &\sim \text{Dirichlet}(\theta_0) \\ w_{d,n} &\sim \text{Multinomial}(\theta_{z_{d,n}}) \end{aligned}$$

Figure 2: The plate diagram for our model. Shaded circles denote observed variables, and squares denote hyperparameters. The dotted arrows indicate that  $\eta$  is constructed deterministically from  $x$  and  $h$ .

ciated with a cluster of keyphrases. At test time, we are presented with documents that do not contain keyphrase annotations. The hidden topic model of the review text is used to determine the properties that a document as a whole supports. For each property, we compute the proportion of the document’s words assigned to it. Properties with proportions above a set threshold (tuned on a development set) are predicted as being supported.

#### 4.1 Keyphrase Clustering

One of our goals is to cluster the keyphrases, such that each cluster corresponds to a well-defined property. We represent each distinct keyphrase as a vector of similarity scores computed over the set of observed keyphrases; these scores are represented by  $s$  in Figure 2, the plate diagram of our model.<sup>1</sup> Modeling the similarity matrix rather than the sur-

<sup>1</sup>We assume that similarity scores are conditionally independent given the keyphrase clustering, though the scores are in fact related. Such simplifying assumptions have been previously used with success in NLP (e.g., Toutanova and Johnson, 2007), though a more theoretically sound treatment of the similarity matrix is an area for future research.

face forms allows arbitrary comparisons between keyphrases, e.g., permitting the use of both lexical and distributional information. The lexical comparison is based on the cosine similarity between the keyphrase words. The distributional similarity is quantified in terms of the co-occurrence of keyphrases across review texts. Our model is inherently capable of using any arbitrary source of similarity information; for a discussion of similarity metrics, see Lin (1998).

#### 4.2 Document-level Distributional Analysis

Our analysis of the document text is based on probabilistic topic models such as LDA (Blei et al., 2003). In the LDA framework, each word is generated from a language model that is indexed by the word’s topic assignment. Thus, rather than identifying a single topic for a document, LDA identifies a distribution over topics.

Our word model operates similarly, identifying a topic for each word, written as  $z$  in Figure 2. To tie these topics to the keyphrases, we deterministically construct a document-specific topic distribu-

tion from the clusters represented by the document’s keyphrases — this is  $\eta$  in the figure.  $\eta$  assigns equal probability to all topics that are represented in the keyphrases, and a small smoothing probability to other topics.

As noted above, properties may be expressed in the text even when no related keyphrase appears. For this reason, we also construct a document-specific topic distribution  $\phi$ . The auxiliary variable  $c$  indicates whether a given word’s topic is drawn from the set of keyphrase clusters, or from this topic distribution.

### 4.3 Generative Process

In this section, we describe the underlying generative process more formally.

First we consider the set of all keyphrases observed across the entire corpus, of which there are  $L$ . We draw a multinomial distribution  $\psi$  over the  $K$  keyphrase clusters from a symmetric Dirichlet prior  $\psi_0$ . Then for the  $\ell^{\text{th}}$  keyphrase, a cluster assignment  $x_\ell$  is drawn from the multinomial  $\psi$ . Finally, the similarity matrix  $\mathbf{s} \in [0, 1]^{L \times L}$  is constructed. Each entry  $s_{\ell, \ell'}$  is drawn independently, depending on the cluster assignments  $x_\ell$  and  $x_{\ell'}$ . Specifically,  $s_{\ell, \ell'}$  is drawn from a Beta distribution with parameters  $\alpha_ =$  if  $x_\ell = x_{\ell'}$  and  $\alpha_{\neq}$  otherwise. The parameters  $\alpha_ =$  linearly bias  $s_{\ell, \ell'}$  towards one (Beta( $\alpha_ =$ )  $\equiv$  Beta(2, 1)), and the parameters  $\alpha_{\neq}$  linearly bias  $s_{\ell, \ell'}$  towards zero (Beta( $\alpha_{\neq}$ )  $\equiv$  Beta(1, 2)).

Next, the words in each of the  $D$  documents are generated. Document  $d$  has  $N_d$  words;  $z_{d,n}$  is the topic for word  $w_{d,n}$ . These latent topics are drawn either from the set of clusters represented by the document’s keyphrases, or from the document’s topic model  $\phi_d$ . We deterministically construct a document-specific keyphrase topic model  $\eta_d$ , based on the keyphrase cluster assignments  $\mathbf{x}$  and the observed keyphrases  $h_d$ . The multinomial  $\eta_d$  assigns equal probability to each topic that is represented by a phrase in  $h_d$ , and a small probability to other topics.

As noted earlier, a document’s text may support properties that are not mentioned in its observed keyphrases. For that reason, we draw a document topic multinomial  $\phi_d$  from a symmetric Dirichlet prior  $\phi_0$ . The binary auxiliary variable  $c_{d,n}$  determines whether the word’s topic is drawn from the

keyphrase model  $\eta_d$  or the document topic model  $\phi_d$ .  $c_{d,n}$  is drawn from a weighted coin flip, with probability  $\lambda$ ;  $\lambda$  is drawn from a Beta distribution with prior  $\lambda_0$ . We have  $z_{d,n} \sim \eta_d$  if  $c_{d,n} = 1$ , and  $z_{d,n} \sim \phi_d$  otherwise. Finally, the word  $w_{d,n}$  is drawn from the multinomial  $\theta_{z_{d,n}}$ , where  $z_{d,n}$  indexes a topic-specific language model. Each of the  $K$  language models  $\theta_k$  is drawn from a symmetric Dirichlet prior  $\theta_0$ .

## 5 Posterior Sampling

Ultimately, we need to compute the model’s posterior distribution given the training data. Doing so analytically is intractable due to the complexity of the model, but sampling-based techniques can be used to estimate the posterior. We employ Gibbs sampling, previously used in NLP by Finkel et al. (2005) and Goldwater et al. (2006), among others. This technique repeatedly samples from the conditional distributions of each hidden variable, eventually converging on a Markov chain whose stationary distribution is the posterior distribution of the hidden variables in the model (Gelman et al., 2004). We now present sampling equations for each of the hidden variables in Figure 2.

The prior over keyphrase clusters  $\psi$  is sampled based on hyperprior  $\psi_0$  and keyphrase cluster assignments  $\mathbf{x}$ . We write  $p(\psi | \dots)$  to mean the probability conditioned on all the other variables.

$$\begin{aligned} p(\psi | \dots) &\propto p(\psi | \psi_0) p(\mathbf{x} | \psi), \\ &= p(\psi | \psi_0) \prod_{\ell} p(x_\ell | \psi) \\ &= \text{Dir}(\psi; \psi_0) \prod_{\ell} \text{Mul}(x_\ell; \psi) \\ &= \text{Dir}(\psi; \psi'), \end{aligned}$$

where  $\psi'_i = \psi_0 + \text{count}(x_\ell = i)$ . This update rule is due to the conjugacy of the multinomial to the Dirichlet distribution. The first line follows from Bayes’ rule, and the second line from the conditional independence of each keyphrase assignment  $x_\ell$  from the others, given  $\psi$ .

$\phi_d$  and  $\theta_k$  are resampled in a similar manner:

$$\begin{aligned} p(\phi_d | \dots) &\propto \text{Dir}(\phi_d; \phi'_d), \\ p(\theta_k | \dots) &\propto \text{Dir}(\theta_k; \theta'_k), \end{aligned}$$

$$\begin{aligned}
p(x_\ell | \dots) &\propto p(x_\ell | \psi) p(\mathbf{s} | x_\ell, \mathbf{x}_{-\ell}, \alpha) p(\mathbf{z} | \eta, \psi, \mathbf{c}) \\
&\propto p(x_\ell | \psi) \left[ \prod_{\ell' \neq \ell} p(s_{\ell, \ell'} | x_\ell, x_{\ell'}, \alpha) \right] \left[ \prod_d \prod_{c_{d,n}=1}^D p(z_{d,n} | \eta_d) \right] \\
&= \text{Mul}(x_\ell; \psi) \left[ \prod_{\ell' \neq \ell} \text{Beta}(s_{\ell, \ell'}; \alpha_{x_\ell, x_{\ell'}}) \right] \left[ \prod_d \prod_{c_{d,n}=1}^D \text{Mul}(z_{d,n}; \eta_d) \right]
\end{aligned}$$

Figure 3: The resampling equation for the keyphrase cluster assignments.

where  $\phi'_{d,i} = \phi_0 + \text{count}(z_{d,n} = i \wedge c_{d,n} = 0)$  and  $\theta'_{k,i} = \theta_0 + \sum_d \text{count}(w_{d,n} = i \wedge z_{d,n} = k)$ . In building the counts for  $\phi'_{d,i}$ , we consider only cases in which  $c_{d,n} = 0$ , indicating that the topic  $z_{d,n}$  is indeed drawn from the document topic model  $\phi_d$ . Similarly, when building the counts for  $\theta'_{k,i}$ , we consider only cases in which the word  $w_{d,n}$  is drawn from topic  $k$ .

To resample  $\lambda$ , we employ the conjugacy of the Beta prior to the Bernoulli observation likelihoods, adding counts of  $c$  to the prior  $\lambda_0$ .

$$p(\lambda | \dots) \propto \text{Beta}(\lambda; \lambda'),$$

$$\text{where } \lambda' = \lambda_0 + \left[ \begin{array}{c} \sum_d \text{count}(c_{d,n} = 1) \\ \sum_d \text{count}(c_{d,n} = 0) \end{array} \right].$$

The keyphrase cluster assignments are represented by  $\mathbf{x}$ , whose sampling distribution depends on  $\psi$ ,  $\mathbf{s}$ , and  $\mathbf{z}$ , via  $\eta$ . The equation is shown in Figure 3. The first term is the prior on  $x_\ell$ . The second term encodes the dependence of the similarity matrix  $\mathbf{s}$  on the cluster assignments; with slight abuse of notation, we write  $\alpha_{x_\ell, x_{\ell'}}$  to denote  $\alpha_{=}$  if  $x_\ell = x_{\ell'}$ , and  $\alpha_{\neq}$  otherwise. The third term is the dependence of the word topics  $z_{d,n}$  on the topic distribution  $\eta_d$ . We compute the final result of Figure 3 for each possible setting of  $x_\ell$ , and then sample from the normalized multinomial.

The word topics  $\mathbf{z}$  are sampled according to keyphrase topic distribution  $\eta_d$ , document topic distribution  $\phi_d$ , words  $\mathbf{w}$ , and auxiliary variables  $\mathbf{c}$ :

$$\begin{aligned}
p(z_{d,n} | \dots) &\propto p(z_{d,n} | \phi_d, \eta_d, c_{d,n}) p(w_{d,n} | z_{d,n}, \theta) \\
&= \begin{cases} \text{Mul}(z_{d,n}; \eta_d) \text{Mul}(w_{d,n}; \theta_{z_{d,n}}) & \text{if } c_{d,n} = 1, \\ \text{Mul}(z_{d,n}; \phi_d) \text{Mul}(w_{d,n}; \theta_{z_{d,n}}) & \text{otherwise.} \end{cases}
\end{aligned}$$

As with  $x_\ell$ , each  $z_{d,n}$  is sampled by computing the conditional likelihood of each possible setting within a constant of proportionality, and then sampling from the normalized multinomial.

Finally, we sample each auxiliary variable  $c_{d,n}$ , which indicates whether the hidden topic  $z_{d,n}$  is drawn from  $\eta_d$  or  $\phi_d$ . The conditional probability for  $c_{d,n}$  depends on its prior  $\lambda$  and the hidden topic assignments  $z_{d,n}$ :

$$\begin{aligned}
p(c_{d,n} | \dots) &\propto p(c_{d,n} | \lambda) p(z_{d,n} | \eta_d, \phi_d, c_{d,n}) \\
&= \begin{cases} \text{Bern}(c_{d,n}; \lambda) \text{Mul}(z_{d,n}; \eta_d) & \text{if } c_{d,n} = 1, \\ \text{Bern}(c_{d,n}; \lambda) \text{Mul}(z_{d,n}; \phi_d) & \text{otherwise.} \end{cases}
\end{aligned}$$

We compute the likelihood of  $c_{d,n} = 0$  and  $c_{d,n} = 1$  within a constant of proportionality, and then sample from the normalized Bernoulli distribution.

## 6 Experimental Setup

**Data Sets** We evaluate our system on reviews from two categories, restaurants and cell phones. These reviews were downloaded from the popular Epinions<sup>2</sup> website. Users of this website evaluate products by providing both a textual description of their opinion, as well as concise lists of keyphrases (pros and cons) summarizing the review. The statistics of this dataset are provided in Table 1. For each of the categories, we randomly selected 50%, 15%, and 35% of the documents as training, development, and test sets, respectively.

Manual analysis of this data reveals that authors often omit properties mentioned in the text from the list of keyphrases. To obtain a complete gold

<sup>2</sup><http://www.epinions.com/>

	Restaurants	Cell Phones
# of reviews	3883	1112
Avg. review length	916.9	1056.9
Avg. keyphrases / review	3.42	4.91

Table 1: Statistics of the reviews dataset by category.

standard, we hand-annotated a subset of the reviews from the restaurant category. The annotation effort focused on eight commonly mentioned properties, such as those underlying the keyphrases “pleasant atmosphere” and “attentive staff.” Two raters annotated 160 reviews, 30 of which were annotated by both. Cohen’s kappa, a measure of interrater agreement ranging from zero to one, was 0.78 for this subset, indicating high agreement (Cohen, 1960).

Each review was annotated with 2.56 properties on average. Each manually-annotated property corresponded to an average of 19.1 keyphrases in the restaurant data, and 6.7 keyphrases in the cell phone data. This supports our intuition that a single semantic property may be expressed using a variety of different keyphrases.

**Training** Our model needs to be provided with the number of clusters  $K$ . We set  $K$  large enough for the model to learn effectively on the development set. For the restaurant data — where the gold standard identified eight semantic properties — we set  $K$  to 20, allowing the model to account for keyphrases not included in the eight most common properties. For the cell phones category, we set  $K$  to 30.

To improve the model’s convergence rate, we perform two initialization steps for the Gibbs sampler. First, sampling is done only on the keyphrase clustering component of the model, ignoring document text. Second, we fix this clustering and sample the remaining model parameters. These two steps are run for 5,000 iterations each. The full joint model is then sampled for 100,000 iterations. Inspection of the parameter estimates confirms model convergence. On a 2GHz dual-core desktop machine, a multi-threaded C++ implementation of model training takes about two hours for each dataset.

**Inference** The final point estimate used for testing is an average (for continuous variables) or a mode (for discrete variables) over the last 1,000 Gibbs sampling iterations. Averaging is a heuristic that is applicable in our case because our sam-

ple histograms are unimodal and exhibit low skew. The model usually works equally well using single-sample estimates, but is more prone to estimation noise.

As previously mentioned, we convert word topic assignments to document properties by examining the proportion of words supporting each property. A threshold for this proportion is set for each property via the development set.

**Evaluation** Our first evaluation examines the accuracy of our model and the baselines by comparing their output against the keyphrases provided by the review authors. More specifically, the model first predicts the properties supported by a given review. We then test whether the original authors’ keyphrases are contained in the clusters associated with these properties.

As noted above, the authors’ keyphrases are often incomplete. To perform a noise-free comparison, we based our second evaluation on the manually constructed gold standard for the restaurant category. We took the most commonly observed keyphrase from each of the eight annotated properties, and tested whether they are supported by the model based on the document text.

In both types of evaluation, we measure the model’s performance using precision, recall, and F-score. These are computed in the standard manner, based on the model’s keyphrase predictions compared against the corresponding references. The sign test was used for statistical significance testing (De Groot and Schervish, 2001).

**Baselines** To the best of our knowledge, this task not been previously addressed in the literature. We therefore consider five baselines that allow us to explore the properties of this task and our model.

*Random:* Each keyphrase is supported by a document with probability of one half. This baseline’s results are computed (in expectation) rather than actually run. This method is expected to have a recall of 0.5, because in expectation it will select half of the correct keyphrases. Its precision is the proportion of supported keyphrases in the test set.

*Phrase in text:* A keyphrase is supported by a document if it appears verbatim in the text. Because of this narrow requirement, precision should be high whereas recall will be low.



	Restaurants gold standard annotation			Restaurants free-text annotation			Cell Phones free-text annotation		
	Recall	Prec.	F-score	Recall	Prec.	F-score	Recall	Prec.	F-score
Random	0.500	0.300	* 0.375	0.500	0.500	* 0.500	0.500	0.489	* 0.494
Phrase in text	0.048	0.500	* 0.087	0.078	0.909	* 0.144	0.171	0.529	* 0.259
Cluster in text	0.223	0.534	0.314	0.517	0.640	* 0.572	0.829	0.547	0.659
Phrase classifier	0.028	0.636	* 0.053	0.068	0.963	* 0.126	0.029	0.600	* 0.055
Cluster classifier	0.113	0.622	◇ 0.192	0.255	0.907	* 0.398	0.210	0.759	0.328
Our model	0.625	0.416	<b>0.500</b>	0.901	0.652	<b>0.757</b>	0.886	0.585	<b>0.705</b>
Our model + gold clusters	0.582	0.398	0.472	0.795	0.627	* 0.701	0.886	0.520	◇ 0.655

Table 2: Comparison of the property predictions made by our model and the baselines in the two categories as evaluated against the gold and free-text annotations. Results for our model using the fixed, manually-created gold clusterings are also shown. The methods against which our model has significantly better results on the sign test are indicated with a \* for  $p \leq 0.05$ , and  $\diamond$  for  $p \leq 0.1$ .

*Cluster in text:* A keyphrase is supported by a document if it or any of its paraphrases appears in the text. Paraphrasing is based on our model’s clustering of the keyphrases. The use of paraphrasing information enhances recall at the potential cost of precision, depending on the quality of the clustering.

*Phrase classifier:* Discriminative classifiers are trained for each keyphrase. Positive examples are documents that are labeled with the keyphrase; all other documents are negative examples. A keyphrase is supported by a document if that keyphrase’s classifier returns positive.

*Cluster classifier:* Discriminative classifiers are trained for each cluster of keyphrases, using our model’s clustering. Positive examples are documents that are labeled with any keyphrase from the cluster; all other documents are negative examples. All keyphrases of a cluster are supported by a document if that cluster’s classifier returns positive.

*Phrase classifier* and *cluster classifier* employ maximum entropy classifiers, trained on the same features as our model, *i.e.*, word counts. The former is high-precision/low-recall, because for any particular keyphrase, its synonymous keyphrases would be considered negative examples. The latter broadens the positive examples, which should improve recall. We used Zhang Le’s MaxEnt toolkit<sup>3</sup> to build these classifiers.

<sup>3</sup>[http://homepages.inf.ed.ac.uk/s0450736/maxent\\_toolkit.html](http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html)

## 7 Results

**Comparative performance** Table 2 presents the results of the evaluation scenarios described above. Our model outperforms every baseline by a wide margin in all evaluations.

The absolute performance of the automatic methods indicates the difficulty of the task. For instance, evaluation against gold standard annotations shows that the random baseline outperforms all of the other baselines. We observe similar disappointing results for the non-random baselines against the free-text annotations. The precision and recall characteristics of the baselines match our previously described expectations.

The poor performance of the discriminative models seems surprising at first. However, these results can be explained by the degree of noise in the training data, specifically, the aforementioned sparsity of free-text annotations. As previously described, our technique allows document text topics to stochastically derive from either the keyphrases or a background distribution — this allows our model to learn effectively from incomplete annotations. In fact, when we force all text topics to derive from keyphrase clusters in our model, its performance degrades to the level of the classifiers or worse, with an F-score of 0.390 in the restaurant category and 0.171 in the cell phone category.

**Impact of paraphrasing** As previously observed in entailment research (Dagan et al., 2006), paraphrasing information contributes greatly to improved performance on semantic inference. This is

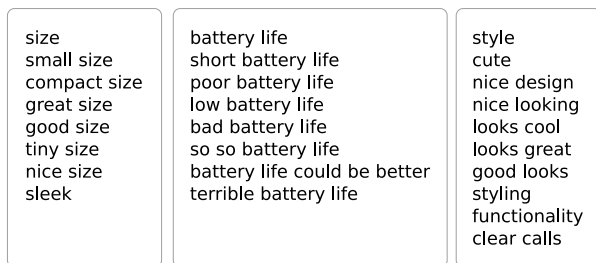


Figure 4: Sample keyphrase clusters that our model infers in the cell phone category.

confirmed by the dramatic difference in results between the *cluster in text* and *phrase in text* baselines. Therefore it is important to quantify the quality of automatically computed paraphrases, such as those illustrated in Figure 4.

	Restaurants	Cell Phones
Keyphrase similarity only	0.931	0.759
Joint training	<b>0.966</b>	<b>0.876</b>

Table 3: Rand Index scores of our model’s clusters, using only keyphrase similarity vs. using keyphrases and text jointly. Comparison of cluster quality is against the gold standard.

One way to assess clustering quality is to compare it against a “gold standard” clustering, as constructed in Section 6. For this purpose, we use the *Rand Index* (Rand, 1971), a measure of cluster similarity. This measure varies from zero to one; higher scores are better. Table 3 shows the Rand Indices for our model’s clustering, as well as the clustering obtained by using only keyphrase similarity. These scores confirm that joint inference produces better clusters than using only keyphrases.

Another way of assessing cluster quality is to consider the impact of using the gold standard clustering instead of our model’s clustering. As shown in the last two lines of Table 2, using the gold clustering yields results worse than using the model clustering. This indicates that for the purposes of our task, the model clustering is of sufficient quality.

## 8 Conclusions and Future Work

In this paper, we have shown how free-text annotations provided by novice users can be leveraged as a training set for document-level semantic inference. The resulting hierarchical Bayesian model

overcomes the lack of consistency in such annotations by inducing a hidden structure of semantic properties, which correspond both to clusters of keyphrases and hidden topic models in the text. Our system successfully extracts semantic properties of unannotated restaurant and cell phone reviews, empirically validating our approach.

Our present model makes strong assumptions about the independence of similarity scores. We believe this could be avoided by modeling the generation of the entire similarity matrix jointly. We have also assumed that the properties themselves are unstructured, but they are in fact related in interesting ways. For example, it would be desirable to model antonyms explicitly, *e.g.*, no restaurant review should be simultaneously labeled as having good and bad food. The correlated topic model (Blei and Lafferty, 2006) is one way to account for relationships between hidden topics; more structured representations, such as hierarchies, may also be considered.

Finally, the core idea of using free-text as a source of training labels has wide applicability, and has the potential to enable sophisticated content search and analysis. For example, online blog entries are often tagged with short keyphrases. Our technique could be used to standardize these tags, and assign keyphrases to untagged blogs. The notion of free-text annotations is also very broad — we are currently exploring the applicability of this model to Wikipedia articles, using section titles as keyphrases, to build standard article schemas.

## Acknowledgments

The authors acknowledge the support of the NSF, Quanta Computer, the U.S. Office of Naval Research, and DARPA. Thanks to Michael Collins, Dina Katabi, Kristian Kersting, Terry Koo, Brian Milch, Tahira Naseem, Dan Roy, Benjamin Snyder, Luke Zettlemoyer, and the anonymous reviewers for helpful comments and suggestions. Any opinions, findings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the NSF.

## References

- David M. Blei and John D. Lafferty. 2006. Correlated topic models. In *Advances in NIPS*, pages 147–154.
- David M. Blei and Jon McAuliffe. 2007. Supervised topic models. In *Advances in NIPS*.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.
- Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. *Lecture Notes in Computer Science*, 3944:177–190.
- Morris H. De Groot and Mark J. Schervish. 2001. *Probability and Statistics*. Addison Wesley.
- Jenny R. Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the ACL*, pages 363–370.
- Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2004. *Bayesian Data Analysis*. Texts in Statistical Science. Chapman & Hall/CRC, 2nd edition.
- Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of ACL*, pages 673–680.
- Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of SIGKDD*, pages 168–177.
- Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL*, pages 483–490.
- Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of ICML*, pages 296–304.
- Ana-Maria Popescu, Bao Nguyen, and Oren Etzioni. 2005. OPINE: Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*, pages 339–346.
- William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, December.
- Bruce Sterling. 2005. Order out of chaos: What is the best way to tag, bag, and sort data? Give it to the unorganized masses. <http://www.wired.com/wired/archive/13.04/view.html?pg=4>. Accessed April 21, 2008.
- Ivan Titov and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the ACL*.
- Kristina Toutanova and Mark Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Advances in NIPS*.
- Graham Vickery and Sacha Wunsch-Vincent. 2007. *Participative Web and User-Created Content: Web 2.0, Wikis and Social Networking*. OECD Publishing.

# Automatic Image Annotation Using Auxiliary Text Information

Yansong Feng and Mirella Lapata

School of Informatics, University of Edinburgh

2 Buccleuch Place, Edinburgh EH8 9LW, UK

Y.Feng-4@sms.ed.ac.uk, mlap@inf.ed.ac.uk

## Abstract

The availability of databases of images labeled with keywords is necessary for developing and evaluating image annotation models. Dataset collection is however a costly and time consuming task. In this paper we exploit the vast resource of images available on the web. We create a database of pictures that are naturally embedded into news articles and propose to use their captions as a proxy for annotation keywords. Experimental results show that an image annotation model can be developed on this dataset alone without the overhead of manual annotation. We also demonstrate that the news article associated with the picture can be used to boost image annotation performance.

## 1 Introduction

As the number of image collections is rapidly growing, so does the need to browse and search them. Recent years have witnessed significant progress in developing methods for image retrieval<sup>1</sup>, many of which are query-based. Given a database of images, each annotated with keywords, the query is used to retrieve relevant pictures under the assumption that the annotations can essentially capture their semantics.

One stumbling block to the widespread use of query-based image retrieval systems is obtaining the keywords for the images. Since manual annotation is expensive, time-consuming and practically infeasible for large databases, there has been great in-

<sup>1</sup>The approaches are too numerous to list; we refer the interested reader to Datta et al. (2005) for an overview.

terest in automating the image annotation process (see references). More formally, given an image  $I$  with visual features  $V_i = \{v_1, v_2, \dots, v_N\}$  and a set of keywords  $W = \{w_1, w_2, \dots, w_M\}$ , the task consists in finding *automatically* the keyword subset  $W_I \subset W$ , which can appropriately describe the image  $I$ . Indeed, several approaches have been proposed to solve this problem under a variety of learning paradigms. These range from supervised classification (Vailaya et al., 2001; Smeulders et al., 2000) to instantiations of the noisy-channel model (Duygulu et al., 2002), to clustering (Barnard et al., 2002), and methods inspired by information retrieval (Lavrenko et al., 2003; Feng et al., 2004).

Obviously in order to develop accurate image annotation models, some manually labeled data is required. Previous approaches have been developed and tested almost exclusively on the Corel database. The latter contains 600 CD-ROMs, each containing about 100 images representing the same topic or concept, e.g., people, landscape, male. Each topic is associated with keywords and these are assumed to also describe the images under this topic. As an example consider the pictures in Figure 1 which are classified under the topic *male* and have the description keywords *man, male, people, cloth, and face*.

Current image annotation methods work well when large amounts of labeled images are available but can run into severe difficulties when the number of images and keywords for a given topic is relatively small. Unfortunately, databases like Corel are few and far between and somewhat idealized. Corel contains clusters of many closely related images which in turn share keyword descriptions, thus allowing models to learn image-keyword associations



Figure 1: Images from the Corel database, exemplifying the concept *male* with keyword descriptions *man*, *male*, *people*, *cloth*, and *face*.

reliably (Tang and Lewis, 2007). It is unlikely that models trained on this database will perform well out-of-domain on other image collections which are more noisy and do not share these characteristics. Furthermore, in order to develop robust image annotation models, it is crucial to have large and diverse datasets both for training and evaluation.

In this work, we aim to relieve the data acquisition bottleneck associated with automatic image annotation by taking advantage of resources where images and their annotations co-occur naturally. News articles associated with images and their captions spring readily to mind (e.g., BBC News, Yahoo News). So, rather than laboriously annotating images with their keywords, we simply treat captions as labels. These annotations are admittedly noisy and far from ideal. Captions can be denotative (describing the objects the image depicts) but also connotative (describing sociological, political, or economic attitudes reflected in the image). Importantly, our images are not standalone, they come with news articles whose content is shared with the image. So, by processing the accompanying document, we can effectively learn about the image and reduce the effect of noise due to the approximate nature of the caption labels. To give a simple example, if two words appear both in the caption and the document, it is more likely that the annotation is genuine.

In what follows, we present a new database consisting of articles, images, and their captions which we collected from an on-line news source. We then propose an image annotation model which can learn from our noisy annotations and the auxiliary documents. Specifically, we extend and modify Lavrenko’s (2003) continuous relevance model

to suit our task. Our experimental results show that this model can successfully scale to our database, without making use of explicit human annotations in any way. We also show that the auxiliary document contains important information for generating more accurate image descriptions.

## 2 Related Work

Automatic image annotation is a popular task in computer vision. The earliest approaches are closely related to image classification (Vailaya et al., 2001; Smeulders et al., 2000), where pictures are assigned a set of simple descriptions such as indoor, outdoor, landscape, people, animal. A binary classifier is trained for each concept, sometimes in a “one vs all” setting. The focus here is mostly on image processing and good feature selection (e.g., colour, texture, contours) rather than the annotation task itself.

Recently, much progress has been made on the image annotation task thanks to three factors. The availability of the Corel database, the use of unsupervised methods and new insights from the related fields of natural language processing and information retrieval. The co-occurrence model (Mori et al., 1999) collects co-occurrence counts between words and image features and uses them to predict annotations for new images. Duygulu et al. (2002) improve on this model by treating image regions and keywords as a bi-text and using the EM algorithm to construct an image region-word dictionary.

Another way of capturing co-occurrence information is to introduce latent variables linking image features with words. Standard latent semantic analysis (LSA) and its probabilistic variant (PLSA) have been applied to this task (Hofmann, 1998). Barnard et al. (2002) propose a hierarchical latent model in order to account for the fact that some words are more general than others. More sophisticated graphical models (Blei and Jordan, 2003) have also been employed including Gaussian Mixture Models (GMM) and Latent Dirichlet Allocation (LDA).

Finally, relevance models originally developed for information retrieval, have been successfully applied to image annotation (Lavrenko et al., 2003; Feng et al., 2004). A key idea behind these models is to find the images most similar to the test image and then use their shared keywords for annotation.

Our approach differs from previous work in two

important respects. Firstly, our ultimate goal is to develop an image annotation model that can cope with real-world images and noisy data sets. To this end we are faced with the challenge of building an appropriate database for testing and training purposes. Our solution is to leverage the vast resource of images available on the web but also the fact that many of these images are implicitly annotated. For example, news articles often contain images whose captions can be thought of as annotations. Secondly, we allow our image annotation model access to knowledge sources other than the image and its keywords. This is relatively straightforward in our case; an image and its accompanying document have shared content, and we can use the latter to glean information about the former. But we hope to illustrate the more general point that auxiliary linguistic information can indeed bring performance improvements on the image annotation task.

### 3 BBC News Database

Our database consists of news images which are abundant. Many on-line news providers supply pictures with news articles, some even classify news into broad topic categories (e.g., business, world, sports, entertainment). Importantly, news images often display several objects and complex scenes and are usually associated with captions describing their contents. The captions are image specific and use a rich vocabulary. This is in marked contrast to the Corel database whose images contain one or two salient objects and a limited vocabulary (typically around 300 words).

We downloaded 3,361 news articles from the BBC News website.<sup>2</sup> Each article was accompanied with an image and its caption. We thus created a database of image-caption-document tuples. The documents cover a wide range of topics including national and international politics, advanced technology, sports, education, etc. An example of an entry in our database is illustrated in Figure 2. Here, the image caption is *Marcin and Florent face intense competition from outside Europe* and the accompanying article discusses EU subsidies to farmers. The images are usually 203 pixels wide and 152 pixels high. The average caption length is 5.35 tokens, and the average document length 133.85 tokens. Our

<sup>2</sup><http://news.bbc.co.uk/>



Figure 2: A sample from our BBC News database. Each entry contains an image, a caption for the image, and the accompanying document with its title.

captions have a vocabulary of 2,167 words and our documents 6,253. The vocabulary shared between captions and documents is 2,056 words.

### 4 Extending the Continuous Relevance Annotation Model

Our work is an extension of the continuous relevance annotation model put forward in Lavrenko et al. (2003). Unlike other unsupervised approaches where a set of latent variables is introduced, each defining a joint distribution on the space of keywords and image features, the relevance model captures the joint probability of images and annotated words *directly*, without requiring an intermediate clustering stage. This model is a good point of departure for our task for several reasons, both theoretical and empirical. Firstly, expectations are computed over every single point in the training set and

therefore parameters can be estimated without EM. Indeed, Lavrenko et al. achieve competitive performance with latent variable models. Secondly, the generation of feature vectors is modeled directly, so there is no need for quantization. Thirdly, as we show below the model can be easily extended to incorporate information outside the image and its keywords.

In the following we first lay out the assumptions underlying our model. We next describe the continuous relevance model in more detail and present our extensions and modifications.

**Assumptions** Since we are using a non-standard database, namely images embedded in documents, it is important to clarify what we mean by image annotation, and how the precise nature of our data impacts the task. We thus make the following assumptions:

1. The caption describes the content of the image directly or indirectly. Unlike traditional image annotation where keywords describe salient objects, captions supply more detailed information, not only about objects, and their attributes, but also events. In Figure 2 the caption mentions Marcin and Florent the two individuals shown in the picture but also the fact that they face competition from outside Europe.
2. Since our images are implicitly rather than explicitly labeled, we do not assume that we can annotate *all* objects present in the image. Instead, we hope to be able to model event-related information such as “what happened”, “who did it”, “when” and “where”. Our annotation task is therefore more semantic in nature than traditionally assumed.
3. The accompanying document describes the content of the image. This is trivially true for news documents where the images conventionally depict events, objects or people mentioned in the article.

To validate these assumptions, we performed the following experiment on our BBC News dataset. We randomly selected 240 image-caption pairs and manually assessed whether the caption content words (i.e., nouns, verbs, and adjectives) could describe the image. We found out that the captions express the picture’s content 90% of the time. Furthermore, approximately 88% of the nouns in sub-

ject or object position directly denote salient picture objects. We thus conclude that the captions contain useful information about the picture and can be used for annotation purposes.

**Model Description** The continuous relevance image annotation model (Lavrenko et al., 2003) generatively learns the joint probability distribution  $P(V, W)$  of words  $W$  and image regions  $V$ . The key assumption here is that the process of generating images is conditionally independent from the process of generating words. Each annotated image in the training set is treated as a latent variable. Then for an unknown image  $I$ , we estimate:

$$P(V_I, W_I) = \sum_{s \in D} P(V_I|s)P(W_I|s)P(s), \quad (1)$$

where  $D$  is the number of images in the training database,  $V_I$  are visual features of the image regions representing  $I$ ,  $W_I$  are the keywords of  $I$ ,  $s$  is a latent variable (i.e., an image-annotation pair), and  $P(s)$  the prior probability of  $s$ . The latter is drawn from a uniform distribution:

$$P(s) = \frac{1}{N_D} \quad (2)$$

where  $N_D$  is number of the latent variables in the training database  $D$ .

When estimating  $P(V_I|s)$ , the probability of image regions and words, Lavrenko et al. (2003) reasonably assume a generative Gaussian kernel distribution for the image regions:

$$P(V_I|s) = \prod_{r=1}^{N_{V_I}} P_g(v_r|s) \quad (3)$$

$$= \prod_{r=1}^{N_{V_I}} \frac{1}{n_{s_v}} \sum_{i=1}^{n_{s_v}} \frac{\exp\{(v_r - v_i)^T \Sigma^{-1} (v_r - v_i)\}}{\sqrt{2^k \pi^k |\Sigma|}}$$

where  $N_{V_I}$  is the number of regions in image  $I$ ,  $v_r$  the feature vector for region  $r$  in image  $I$ ,  $n_{s_v}$  the number of regions in the image of latent variable  $s$ ,  $v_i$  the feature vector for region  $i$  in  $s$ ’s image,  $k$  the dimension of the image feature vectors and  $\Sigma$  the feature covariance matrix. According to equation (3), a Gaussian kernel is fit to every feature vector  $v_i$  corresponding to region  $i$  in the image of the latent variable  $s$ . Each kernel here is determined by the feature covariance matrix  $\Sigma$ , and for simplicity,  $\Sigma$  is assumed to be a diagonal matrix:  $\Sigma = \beta I$ , where  $I$  is the identity matrix; and  $\beta$  is a scalar modulating the bandwidth of

the kernel whose value is optimized on the development set.

Lavrenko et al. (2003) estimate the word probabilities  $P(W|s)$  using a multinomial distribution. This is a reasonable assumption in the Corel dataset, where the annotations have similar lengths and the words reflect the salience of objects in the image (the multinomial model tends to favor words that appear multiple times in the annotation). However, in our dataset the annotations have varying lengths, and do not necessarily reflect object salience. We are more interested in modeling the *presence* or *absence* of words in the annotation and thus use the multiple-Bernoulli distribution to generate words (Feng et al., 2004). And rather than relying solely on annotations in the training database, we can also take the accompanying document into account using a weighted combination.

The probability of sampling a set of words  $W$  given a latent variable  $s$  from the underlying multiple Bernoulli distribution that has generated the training set  $D$  is:

$$P(W|s) = \prod_{w \in W} P(w|s) \prod_{w \notin W} (1 - P(w|s)) \quad (4)$$

where  $P(w|s)$  denotes the probability of the  $w$ 'th component of the multiple Bernoulli distribution. Now, in estimating  $P(w|s)$  we can include the document as:

$$P_{est}(w|s) = \alpha P_{est}(w|s_a) + (1 - \alpha) P_{est}(w|s_d) \quad (5)$$

where  $\alpha$  is a smoothing parameter tuned on the development set,  $s_a$  is the annotation for the latent variable  $s$  and  $s_d$  its corresponding document.

Equation (5) smooths the influence of the annotation words and allows to offset the negative effect of the noise inherent in our dataset. Since our images are implicitly annotated, there is no guarantee that the annotations are all appropriate. By taking into account  $P_{est}(w|s_d)$ , it is possible to annotate an image with a word that appears in the document but is not included in the caption.

We use a Bayesian framework for estimating  $P_{est}(w|s_a)$ . Specifically, we assume a beta prior (conjugate to the Bernoulli distribution) for each word:

$$P_{est}(w|s_a) = \frac{\mu b_{w,s_a} + N_w}{\mu + D} \quad (6)$$

where  $\mu$  is a smoothing parameter estimated on the development set,  $b_{w,s_a}$  is a Boolean variable denoting whether  $w$  appears in the annotation  $s_a$ , and  $N_w$  is the number of latent variables that contain  $w$  in their annotations.

We estimate  $P_{est}(w|s_d)$  using maximum likelihood estimation (Ponte and Croft, 1998):

$$P_{est}(w|s_d) = \frac{num_{w,s_d}}{num_{s_d}} \quad (7)$$

where  $num_{w,s_d}$  denotes the frequency of  $w$  in the accompanying document of latent variable  $s$  and  $num_{s_d}$  the number of all tokens in the document. Note that we purposely leave  $P_{est}$  unsmoothed, since it is used as a means of balancing the weight of word frequencies in annotations. So, if a word does not appear in the document, the possibility of selecting it will not be greater than  $\alpha$  (see Equation (5)).

Unfortunately, including the document in the estimation of  $P_{est}(w|s)$  increases the vocabulary which in turn increases computation time. Given a test image-document pair, we must evaluate  $P(w|V_I)$  for every  $w$  in our vocabulary which is the union of the caption and document words. We reduce the search space, by scoring each document word with its  $tf * idf$  weight (Salton and McGill, 1983) and adding the  $n$ -best candidates to our caption vocabulary. This way the vocabulary is not fixed in advance for all images but changes dynamically depending on the document at hand.

**Re-ranking the Annotation Hypotheses** It is easy to see that the output of our model is a ranked word list. Typically, the  $k$ -best words are taken to be the automatic annotations for a test image  $I$  (Duygulu et al., 2002; Lavrenko et al., 2003; Jeon and Manmatha, 2004) where  $k$  is a small number and the same for all images.

So far we have taken account of the auxiliary document rather naively, by considering its vocabulary in the estimation of  $P(W|s)$ . Crucially, documents are written with one or more topics in mind. The image (and its annotations) are likely to represent these topics, so ideally our model should prefer words that are strong topic indicators. A simple way to implement this idea is by re-ranking our  $k$ -best list according to a topic model estimated from the entire document collection.

Specifically, we use Latent Dirichlet Allocation (LDA) as our topic model (Blei et al., 2003). LDA



represents documents as a mixture of topics and has been previously used to perform document classification (Blei et al., 2003) and ad-hoc information retrieval (Wei and Croft, 2006) with good results. Given a collection of documents and a set of latent variables (i.e., the number of topics), the LDA model estimates the probability of topics per document and the probability of words per topic. The topic mixture is drawn from a conjugate Dirichlet prior that remains the same for all documents.

For our re-ranking task, we use the LDA model to infer the  $m$ -best topics in the accompanying document. We then select from the output of our model those words that are most likely according to these topics. To give a concrete example, let us assume that for a given image our model has produced five annotations,  $w_1, w_2, w_3, w_4$ , and  $w_5$ . However, according to the LDA model neither  $w_2$  nor  $w_5$  are likely topic indicators. We therefore remove  $w_2$  and  $w_5$  and substitute them with words further down the ranked list that are topical (e.g.,  $w_6$  and  $w_7$ ). An advantage of using LDA is that at test time we can perform inference without retraining the topic model.

## 5 Experimental Setup

In this section we discuss our experimental design for assessing the performance of the model presented above. We give details on our training procedure and parameter estimation, describe our features, and present the baseline methods used for comparison with our approach.

**Data** Our model was trained and tested on the database introduced in Section 3. We used 2,881 image-caption-document tuples for training, 240 tuples for development and 240 for testing. The documents and captions were part-of-speech tagged and lemmatized with Tree Tagger (Schmid, 1994). Words other than nouns, verbs, and adjectives were discarded. Words that were attested less than five times in the training set were also removed to avoid unreliable estimation. In total, our vocabulary consisted of 8,309 words.

**Model Parameters** Images are typically segmented into regions prior to training. We impose a fixed-size rectangular grid on each image rather than attempting segmentation using a general purpose algorithm such as normalized cuts (Shi and Malik,

Color
average of RGB components, standard deviation
average of LUV components, standard deviation
average of LAB components, standard deviation
Texture
output of DCT transformation
output of Gabor filtering (4 directions, 3 scales)
Shape
oriented edge (4 directions)
ratio of edge to non-edge

Table 2: Set of image features used in our experiments.

2000). Using a grid avoids unnecessary errors from image segmentation algorithms, reduces computation time, and simplifies parameter estimation (Feng et al., 2004). Taking the small size and low resolution of the BBC News images into account, we average divide each image into  $6 \times 5$  rectangles and extract features for each region. We use 46 features based on color, texture, and shape. They are summarized in Table 2.

The model presented in Section 4 has a few parameters that must be selected empirically on the development set. These include the vocabulary size, which is dependent on the  $n$  words with the highest  $tf * idf$  scores in each document, and the number of topics for the LDA-based re-ranker. We obtained best performance with  $n$  set to 100 (no cutoff was applied in cases where the vocabulary was less than 100). We trained an LDA model with 20 topics on our document collection using David Blei’s implementation.<sup>3</sup> We used this model to re-rank the output of our annotation model according to the three most likely topics in each document.

**Baselines** We compared our model against three baselines. The first baseline is based on  $tf * idf$  (Salton and McGill, 1983). We rank the document’s content words (i.e., nouns, verbs, and adjectives) according to their  $tf * idf$  weight and select the top  $k$  to be the final annotations. Our second baseline simply annotates the image with the document’s title. Again we only use content words (the average title length in the training set was 4.0 words). Our third baseline is Lavrenko et al.’s (2003) continuous relevance model. It is trained solely on image-caption

<sup>3</sup>Available from <http://www.cs.princeton.edu/~blei/lda-c/index.html>.

Model	Top 10			Top 15			Top 20		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
<i>tf * idf</i>	4.37	7.09	5.41	3.57	8.12	4.86	2.65	8.89	4.00
DocTitle	9.22	7.03	7.20	9.22	7.03	7.20	9.22	7.03	7.20
Lavrenko03	9.05	16.01	11.81	7.73	17.87	10.71	6.55	19.38	9.79
ExtModel	14.72	27.95	19.82	11.62	32.99	17.18	9.72	36.77	15.39

Table 1: Automatic image annotation results on the BBC News database.

pairs, uses a vocabulary of 2,167 words and the same features as our extended model.

**Evaluation** Our evaluation follows the experimental methodology proposed in Duygulu et al. (2002). We are given an un-annotated image  $I$  and are asked to automatically produce suitable annotations for  $I$ . Given a set of image regions  $V_I$ , we use equation (1) to derive the conditional distribution  $P(w|V_I)$ . We consider the  $k$ -best words as the annotations for  $I$ . We present results using the top 10, 15, and 20 annotation words. We assess our model’s performance using precision/recall and F1. In our task, precision is the percentage of correctly annotated words over all annotations that the system suggested. Recall, is the percentage of correctly annotated words over the number of genuine annotations in the test data. F1 is the harmonic mean of precision and recall. These measures are averaged over the set of test words.

## 6 Results

Our experiments were driven by three questions: (1) Is it possible to create an annotation model from noisy data that has not been explicitly hand labeled for this task? (2) What is the contribution of the auxiliary document? As mentioned earlier, considering the document increases the model’s computational complexity, which can be justified as long as we demonstrate a substantial increase in performance. (3) What is the contribution of the image? Here, we are trying to assess if the image features matter. For instance, we could simply generate annotation words by processing the document alone.

Our results are summarized in Table 1. We compare the annotation performance of the model proposed in this paper (ExtModel) with Lavrenko et al.’s (2003) original continuous relevance model (Lavrenko03) and two other simpler models which

do not take the image into account (*tf \* idf* and DocTitle). First, note that the original relevance model performs best when the annotation output is restricted to 10 words with an F1 of 11.81% (recall is 9.05 and precision 16.01). F1 is marginally worse with 15 output words and decreases by 2% with 20. This model does not take any document-based information into account, it is trained solely on image-caption pairs. On the Corel test set the same model obtains a precision of 19.0% and a recall of 16.0% with a vocabulary of 260 words. Although these results are not strictly comparable with ours due to the different nature of the training data (in addition, we output 10 annotation words, whereas Lavrenko et al. (2003) output 5), they give some indication of the decrease in performance incurred when using a more challenging dataset. Unlike Corel, our images have greater variety, non-overlapping content and employ a larger vocabulary (2,167 vs. 260 words).

When the document is taken into account (see ExtModel in Table 1), F1 improves by 8.01% (recall is 14.72% and precision 27.95%). Increasing the size of the output annotations to 15 or 20 yields better recall, at the expense of precision. Eliminating the LDA reranker from the extended model decreases F1 by 0.62%. Incidentally, LDA can be also used to rerank the output of Lavrenko et al.’s (2003) model. LDA also increases the performance of this model by 0.41%.

Finally, considering the document alone, without the image yields inferior performance. This is true for the *tf \* idf* model and the model based on the document titles.<sup>4</sup> Interestingly, the latter yields precision similar to Lavrenko et al. (2003). This is probably due to the fact that the document’s title is in a sense similar to a caption. It often contains words that describe the document’s gist and expectedly

<sup>4</sup>Reranking the output of these models with LDA slightly decreases performance (approximately by 0.2%).




			
<i>tf * idf</i>	<b>breastfeed</b> , medical, intelligent, health, <u>child</u>	culturalism, faith, Muslim, separateness, ethnic	<b>ceasefire</b> , <u>Lebanese</u> , disarm, cabinet, Haaretz
DocTitle	Breast milk does not boost IQ	UK must tackle ethnic tensions	Mid-East hope as ceasefire begins
Lavrenko03	woman, <b>baby</b> , hospital, new, day, lead, good, England, look, family	bomb, city, want, day, <u>fight</u> , child, <u>attack</u> , face, help, government	war, carry, city, security, <b>Israeli</b> , attack, minister, <u>force</u> , government, leader
ExtModel	<b>breastfeed</b> , intelligent, <b>baby</b> , mother, <b>tend</b> , <u>child</u> , study, woman, sibling, advantage	aim, Kelly, faith, culturalism, community, Ms, tension, commission, multi, tackle, school	<b>Lebanon</b> , <b>Israeli</b> , <u>Lebanese</u> , aeroplane, <b>troop</b> , <u>Hezbollah</u> , Israel, <u>force</u> , <b>ceasefire</b> , grey
Caption	Breastfed babies tend to be brighter	Segregation problems were blamed for 2001's Bradford riots	Thousands of Israeli troops are in Lebanon as the ceasefire begins

Figure 3: Examples of annotations generated by our model (ExtModel), the continuous relevance model (Lavrenko03), and the two baselines based on *tf \* idf* and the document title (DocTitle). Words in bold face indicate exact matches, underlined words are semantically compatible. The original captions are in the last row.

some of these words will be also appropriate for the image. In fact, in our dataset, the title words are a subset of those found in the captions.

Examples of the annotations generated by our model are shown in Figure 3. We also include the annotations produced by Lavrenko et. al's (2003) model and the two baselines. As we can see our model annotates the image with words that are not always included in the caption. Some of these are synonyms of the caption words (e.g., *child* and *intelligent* in left image of Figure 3), whereas others express additional information (e.g., *mother*, *woman*). Also note that complex scene images remain challenging (see the center image in Figure 3). Such images are better analyzed at a higher resolution and probably require more training examples.

## 7 Conclusions and Future Work

In this paper, we describe a new approach for the collection of image annotation datasets. Specifically, we leverage the vast resource of images available on the Internet while exploiting the fact that many of them are labeled with captions. Our experiments show that it is possible to learn an image annotation model from caption-picture pairs even if these are not explicitly annotated in any way. We also show that the annotation model benefits substantially from

additional information, beyond the caption or image. In our case this information is provided by the news documents associated with the pictures. But more generally our results indicate that further linguistic knowledge is needed to improve performance on the image annotation task. For instance, resources like WordNet (Fellbaum, 1998) can be used to expand the annotations by exploiting information about is-a relationships.

The uses of the database discussed in this article are many and varied. An interesting future direction concerns the application of the proposed model in a semi-supervised setting where the annotation output is iteratively refined with some manual intervention. Another possibility would be to use the document to increase the annotation keywords by identifying synonyms or even sentences that are similar to the image caption. Also note that our analysis of the accompanying document was rather shallow, limited to part of speech tagging. It is reasonable to assume that results would improve with more sophisticated preprocessing (i.e., named entity recognition, parsing, word sense disambiguation). Finally, we also believe that the model proposed here can be usefully employed in an information retrieval setting, where the goal is to find the image most relevant for a given query or document.

## References

- K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. Blei, and M. Jordan. 2002. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135.
- D. Blei and M. Jordan. 2003. Modeling annotated data. In *Proceedings of the 26th Annual International ACM SIGIR Conference*, pages 127–134, Toronto, ON.
- D. Blei, A. Ng, and M. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.
- R. Datta, J. Li, and J. Z. Wang. 2005. Content-based image retrieval – approaches and trends of the new age. In *Proceedings of the International Workshop on Multimedia Information Retrieval*, pages 253–262, Singapore.
- P. Duygulu, K. Barnard, J. de Freitas, and D. Forsyth. 2002. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the 7th European Conference on Computer Vision*, pages 97–112, Copenhagen, Denmark.
- C. Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.
- S. Feng, V. Lavrenko, and R. Manmatha. 2004. Multiple Bernoulli relevance models for image and video annotation. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 1002–1009, Washington, DC.
- T. Hofmann. 1998. Learning and representing topic. A hierarchical mixture model for word occurrences in document databases. In *Proceedings of the Conference for Automated Learning and Discovery*, pages 408–415, Pittsburgh, PA.
- J. Jeon and R. Manmatha. 2004. Using maximum entropy for automatic image annotation. In *Proceedings of the 3rd International Conference on Image and Video Retrieval*, pages 24–32, Dublin City, Ireland.
- V. Lavrenko, R. Manmatha, and J. Jeon. 2003. A model for learning the semantics of pictures. In *Proceedings of the 16th Conference on Advances in Neural Information Processing Systems*, Vancouver, BC.
- Y. Mori, H. Takahashi, and R. Oka. 1999. Image-to-word transformation based on dividing and vector quantizing images with words. In *Proceedings of the 1st International Workshop on Multimedia Intelligent Storage and Retrieval Management*, Orlando, FL.
- J. M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference*, pages 275–281, New York, NY.
- G. Salton and M.J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.
- H. Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.
- J. Shi and J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.
- A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. 2000. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380.
- J. Tang and P. H. Lewis. 2007. A study of quality issues for image auto-annotation with the Corel data-set. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(3):384–389.
- A. Vailaya, M. Figueiredo, A. Jain, and H. Zhang. 2001. Image classification for content-based indexing. *IEEE Transactions on Image Processing*, 10:117–130.
- X. Wei and B. W. Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference*, pages 178–185, Seattle, WA.

# Hedge classification in biomedical texts with a weakly supervised selection of keywords

György Szarvas

Research Group on Artificial Intelligence  
Hungarian Academy of Sciences / University of Szeged  
HU-6720 Szeged, Hungary  
szarvas@inf.u-szeged.hu

## Abstract

Since facts or statements in a hedge or negated context typically appear as false positives, the proper handling of these language phenomena is of great importance in biomedical text mining. In this paper we demonstrate the importance of hedge classification experimentally in two real life scenarios, namely the ICD-9-CM coding of radiology reports and gene name Entity Extraction from scientific texts. We analysed the major differences of speculative language in these tasks and developed a maxent-based solution for both the free text and scientific text processing tasks. Based on our results, we draw conclusions on the possible ways of tackling speculative language in biomedical texts.

## 1 Introduction

The highly accurate identification of several regularly occurring language phenomena like the speculative use of language, negation and past tense (temporal resolution) is a prerequisite for the efficient processing of biomedical texts. In various natural language processing tasks, relevant statements appearing in a speculative context are treated as false positives. Hedge detection seeks to perform a kind of semantic filtering of texts, that is it tries to separate factual statements from speculative/uncertain ones.

### 1.1 Hedging in biomedical NLP

To demonstrate the detrimental effects of speculative language on biomedical NLP tasks, we will consider two inherently different sample tasks, namely

the ICD-9-CM coding of radiology records and gene information extraction from biomedical scientific texts. The general features of texts used in these tasks differ significantly from each other, but both tasks require the exclusion of uncertain (or speculative) items from processing.

#### 1.1.1 Gene Name and interaction extraction from scientific texts

The test set of the hedge classification dataset<sup>1</sup> (Medlock and Briscoe, 2007) has also been annotated for gene names<sup>2</sup>.

Examples of speculative assertions:

*Thus, the D-mib wing phenotype may result from defective N inductive signaling at the D-V boundary. A similar role of Croquemort has not yet been tested, but seems likely since the crq mutant used in this study (crqKG01679) is lethal in pupae.*

After an automatic parallelisation of the 2 annotations (sentence matching) we found that a significant part of the gene names mentioned (638 occurrences out of a total of 1968) appears in a speculative sentence. This means that approximately 1 in every 3 genes should be excluded from the interaction detection process. These results suggest that a major portion of system false positives could be due to hedging if hedge detection had been neglected by a gene interaction extraction system.

#### 1.1.2 ICD-9-CM coding of radiology records

Automating the assignment of ICD-9-CM codes for radiology records was the subject of a shared task

<sup>1</sup><http://www.cl.cam.ac.uk/~bwm23/>

<sup>2</sup><http://www.cl.cam.ac.uk/~nk304/>

challenge organised in Spring 2007. The detailed description of the task, and the challenge itself can be found in (Pestian et al., 2007) and online<sup>3</sup>. ICD-9-CM codes that are assigned to each report after the patient’s clinical treatment are used for the reimbursement process by insurance companies. There are official guidelines for coding radiology reports (Moisio, 2006). These guidelines strictly state that an uncertain diagnosis should never be coded, hence identifying reports with a diagnosis in a speculative context is an inevitable step in the development of automated ICD-9-CM coding systems. The following examples illustrate a typical non-speculative context where a given code should be added, and a speculative context where the same code should never be assigned to the report:

**non-speculative:** *Subsegmental **atelectasis** in the left lower lobe, otherwise normal exam.*

**speculative:** *Findings suggesting viral or reactive airway disease with right lower lobe **atelectasis** or pneumonia.* In an ICD-9 coding system developed for the challenge, the inclusion of a hedge classifier module (a simple keyword-based lookup method with 38 keywords) improved the overall system performance from 79.7% to 89.3%.

## 1.2 Related work

Although a fair amount of literature on hedging in scientific texts has been produced since the 1990s (e.g. (Hyland, 1994)), speculative language from a Natural Language Processing perspective has only been studied in the past few years. This phenomenon, together with others used to express forms of authorial opinion, is often classified under the notion of subjectivity (Wiebe et al., 2004), (Shanahan et al., 2005). Previous studies (Light et al., 2004) showed that the detection of hedging can be solved effectively by looking for specific keywords which imply that the content of a sentence is speculative and constructing simple expert rules that describe the circumstances of where and how a keyword should appear. Another possibility is to treat the problem as a classification task and train a statistical model to discriminate speculative and non-speculative assertions. This approach requires the availability of labeled instances to train the models

on. Riloff et al. (Riloff et al., 2003) applied bootstrapping to recognise subjective noun keywords and classify sentences as subjective or objective in newswire texts. Medlock and Briscoe (Medlock and Briscoe, 2007) proposed a weakly supervised setting for hedge classification in scientific texts where the aim is to minimise human supervision needed to obtain an adequate amount of training data.

Here we follow (Medlock and Briscoe, 2007) and treat the identification of speculative language as the classification of sentences for either speculative or non-speculative assertions, and extend their methodology in several ways. Thus given labeled sets  $S_{spec}$  and  $S_{nspec}$  the task is to train a model that, for each sentence  $s$ , is capable of deciding whether a previously unseen  $s$  is speculative or not.

The contributions of this paper are the following:

- The construction of a complex feature selection procedure which successfully reduces the number of keyword candidates without excluding helpful keywords.
- We demonstrate that with a very limited amount of expert supervision in finalising the feature representation, it is possible to build accurate hedge classifiers from (semi-) automatically collected training data.
- The extension of the feature representation used by previous works with bigrams and trigrams and an evaluation of the benefit of using longer keywords in hedge classification.
- We annotated a small test corpora of biomedical scientific papers from a different source to demonstrate that hedge keywords are highly task-specific and thus constructing models that generalise well from one task to another is not feasible without a noticeable loss in accuracy.

## 2 Methods

### 2.1 Feature space representation

Hedge classification can essentially be handled by acquiring task specific keywords that trigger speculative assertions more or less independently of each other. As regards the nature of this task, a vector space model (VSM) is a straightforward and suitable representation for statistical learning. As VSM

<sup>3</sup><http://www.computationalmedicine.org/challenge/index.php>

is inadequate for capturing the (possibly relevant) relations between subsequent tokens, we decided to extend the representation with bi- and trigrams of words. We chose not to add any weighting of features (by frequency or importance) and for the Maximum Entropy Model classifier we included binary data about whether single features occurred in the given context or not.

## 2.2 Probabilistic training data acquisition

To build our classifier models, we used the dataset gathered and made available by (Medlock and Briscoe, 2007). They commenced with the seed set  $S_{spec}$  gathered automatically (all sentences containing *suggest* or *likely* – two very good speculative keywords), and  $S_{nspec}$  that consisted of randomly selected sentences from which the most probable speculative instances were filtered out by a pattern matching and manual supervision procedure. With these seed sets they then performed the following iterative method to enlarge the initial training sets, adding examples to both classes from an unlabelled pool of sentences called  $U$ :

1. Generate seed training data:  $S_{spec}$  and  $S_{nspec}$
2. Initialise:  $T_{spec} \leftarrow S_{spec}$  and  $T_{nspec} \leftarrow S_{nspec}$
3. Iterate:
  - Train classifier using  $T_{spec}$  and  $T_{nspec}$
  - Order  $U$  by  $P(spec)$  values assigned by the classifier
  - $T_{spec} \leftarrow$  most probable batch
  - $T_{nspec} \leftarrow$  least probable batch

What makes this iterative method efficient is that, as we said earlier, hedging is expressed via keywords in natural language texts; and often several keywords are present in a single sentence. The seed set  $S_{spec}$  contained either *suggest* or *likely*, and due to the fact that other keywords cooccur with these two in many sentences, they appeared in  $S_{spec}$  with reasonable frequency. For example,  $P(spec|may) = 0.9985$  on the seed sets created by (Medlock and Briscoe, 2007). The iterative extension of the training sets for each class further boosted this effect, and skewed the distribution of speculative indicators as sentences containing them

were likely to be added to the extended training set for the speculative class, and unlikely to fall into the non-speculative set.

We should add here that the very same feature has an inevitable, but very important side effect that is detrimental to the classification accuracy of models trained on a dataset which has been obtained this way. This side effect is that other words (often common words or stopwords) that tend to cooccur with hedge cues will also be subject to the same iterative distortion of their distribution in speculative and non-speculative uses. Perhaps the best example of this is the word *it*. Being a stopword in our case, and having no relevance at all to speculative assertions, it has a class conditional probability of  $P(spec|it) = 74.67\%$  on the seed sets. This is due to the use of phrases like *it suggests that*, *it is likely*, and so on. After the iterative extension of training sets, the class-conditional probability of *it* dramatically increased, to  $P(spec|it) = 94.32\%$ . This is a consequence of the frequent co-occurrence of *it* with meaningful hedge cues and the probabilistic model used and happens with many other irrelevant terms (not just stopwords). The automatic elimination of these irrelevant candidates is one of our main goals (to limit the number of candidates for manual consideration and thus to reduce the human effort required to select meaningful hedge cues).

This shows that, in addition to the desired effect of introducing further speculative keywords and biasing their distribution towards the speculative class, this iterative process also introduces significant noise into the dataset. This observation led us to the conclusion that in order to build efficient classifiers based on this kind of dataset, we should filter out noise. In the next part we will present our feature selection procedure (evaluated in the Results section) which is capable of underranking irrelevant keywords in the majority of cases.

## 2.3 Feature (or keyword) selection

To handle the inherent noise in the training dataset that originates from its weakly supervised construction, we applied the following feature selection procedure. The main idea behind it is that it is unlikely that more than two keywords are present in the text, which are useful for deciding whether an instance is speculative. Here we performed the following steps:

1. We ranked the features  $x$  by frequency and their class conditional probability  $P(spec|x)$ . We then selected those features that had  $P(spec|x) > 0.94$  (this threshold was chosen arbitrarily) and appeared in the training dataset with reasonable frequency (frequency above  $10^{-5}$ ). This set constituted the 2407 candidates which we used in the second analysis phase.
2. For trigrams, bigrams and unigrams – processed separately – we calculated a new class-conditional probability for each feature  $x$ , discarding those observations of  $x$  in speculative instances where  $x$  was not among the two highest ranked candidate. Negative credit was given for all occurrences in non-speculative contexts. We discarded any feature that became unreliable (i.e. any whose frequency dropped below the threshold or the strict class-conditional probability dropped below 0.94). We did this separately for the uni-, bi- and trigrams to avoid filtering out longer phrases because more frequent, shorter candidates took the credit for all their occurrences. In this step we filtered out 85% of all the keyword candidates and kept 362 uni-, bi-, and trigrams altogether.
3. In the next step we re-evaluated all 362 candidates together and filtered out all phrases that had a shorter and thus more frequent substring of themselves among the features, with a similar class-conditional probability on the speculative class (worse by 2% at most). Here we discarded a further 30% of the candidates and kept 253 uni-, bi-, and trigrams altogether.

This efficient way of reranking and selecting potentially relevant features (we managed to discard 89.5% of all the initial candidates automatically) made it easier for us to manually validate the remaining keywords. This allowed us to incorporate supervision into the learning model in the feature representation stage, but keep the weakly supervised modelling (with only 5 minutes of expert supervision required).

## 2.4 Maximum Entropy Classifier

Maximum Entropy Models (Berger et al., 1996) seek to maximise the conditional probability of classes, given certain observations (features). This is performed by weighting features to maximise the likelihood of data and, for each instance, decisions are made based on features present at that point, thus maxent classification is quite suitable for our purposes. As feature weights are mutually estimated, the maxent classifier is capable of taking feature dependence into account. This is useful in cases like the feature  $it$  being dependent on others when observed in a speculative context. By downweighting such features, maxent is capable of modelling to a certain extent the special characteristics which arise from the automatic or weakly supervised training data acquisition procedure. We used the OpenNLP maxent package, which is freely available<sup>4</sup>.

## 3 Results

In this section we will present our results for hedge classification as a standalone task. In experiments we made use of the hedge classification dataset of scientific texts provided by (Medlock and Briscoe, 2007) and used a labeled dataset generated automatically based on false positive predictions of an ICD-9-CM coding system.

### 3.1 Results for hedge classification in biomedical texts

As regards the degree of human intervention needed, our classification and feature selection model falls within the category of weakly supervised machine learning. In the following sections we will evaluate our above-mentioned contributions one by one, describing their effects on feature space size (efficiency in feature and noise filtering) and classification accuracy. In order to compare our results with Medlock and Briscoe’s results (Medlock and Briscoe, 2007), we will always give the  $BEP(spec)$  that they used – the break-even-point of precision and recall<sup>5</sup>. We will also present  $F_{\beta=1}(spec)$  values

<sup>4</sup><http://maxent.sourceforge.net/>

<sup>5</sup>It is the point on the precision-recall curve of  $spec$  class where  $P = R$ . If an exact  $P = R$  cannot be realised due to the equal ranking of many instances, we use the point closest to  $P = R$  and set  $BEP(spec) = (P + R)/2$ . BEP is an



which show how good the models are at recognising speculative assertions.

### 3.1.1 The effects of automatic feature selection

The method we proposed seems especially effective in the sense that we successfully reduced the number of keyword candidates from an initial 2407 words having  $P(spec|x) > 0.94$  to 253, which is a reduction of almost 90%. During the process, very few useful keywords were eliminated and this indicated that our feature selection procedure was capable of distinguishing useful keywords from noise (i.e. keywords having a very high speculative class-conditional probability due to the skewed characteristics of the automatically gathered training dataset). The 2407-keyword model achieved a  $BEP(spec)$  of 76.05% and  $F_{\beta=1}(spec)$  of 73.61%, while the model after feature selection performed better, achieving a  $BEP(spec)$  score of 78.68% and  $F_{\beta=1}(spec)$  score of 78.09%. Simplifying the model to predict a *spec* label each time a keyword was present (by discarding those 29 features that were too weak to predict *spec* alone) slightly increased both the  $BEP(spec)$  and  $F_{\beta=1}(spec)$  values to 78.95% and 78.25%. This shows that the Maximum Entropy Model in this situation could not learn any meaningful hypothesis from the co-occurrence of individually weak keywords.

### 3.1.2 Improvements by manual feature selection

After a dimension reduction via a strict reranking of features, the resulting number of keyword candidates allowed us to sort the retained phrases manually and discard clearly irrelevant ones. We judged a phrase irrelevant if we could consider no situation in which the phrase could be used to express hedging. Here 63 out of the 253 keywords retained by the automatic selection were found to be **potentially** relevant in hedge classification. All these features were sufficient for predicting the *spec* class alone, thus we again found that the learnt model reduced to a single keyword-based decision.<sup>6</sup> These 63 key-

interesting metric as it demonstrates how well we can trade-off precision for recall.

<sup>6</sup>We kept the test set blind during the selection of relevant keywords. This meant that some of them eventually proved to be irrelevant, or even lowered the classification accuracy. Examples of such keywords were *will*, *these data* and *hypothesis*.

words yielded a classifier with a  $BEP(spec)$  score of 82.02% and  $F_{\beta=1}(spec)$  of 80.88%.

### 3.1.3 Results obtained adding external dictionaries

In our final model we added the keywords used in (Light et al., 2004) and those gathered for our ICD-9-CM hedge detection module. Here we decided not to check whether these keywords made sense in scientific texts or not, but instead left this task to the maximum entropy classifier, and added only those keywords that were found reliable enough to predict *spec* label alone by the maxent model trained on the training dataset. These experiments confirmed that hedge cues are indeed task specific – several cues that were reliable in radiology reports proved to be of no use for scientific texts. We managed to increase the number of our features from 63 to 71 using these two external dictionaries.

These additional keywords helped us to increase the overall coverage of the model. Our final hedge classifier yielded a  $BEP(spec)$  score of 85.29% and  $F_{\beta=1}(spec)$  score of 85.08% (89.53% Precision, 81.05% Recall) for the speculative class. This meant an overall classification accuracy of 92.97%.

Using this system as a pre-processing module for a hypothetical gene interaction extraction system, we found that our classifier successfully excluded gene names mentioned in a speculative sentence (it removed 81.66% of all speculative mentions) and this filtering was performed with a respectable precision of 93.71% ( $F_{\beta=1}(spec) = 87.27%$ ).

Articles	4
Sentences	1087
Spec sentences	190
Nspec sentences	897

Table 1: Characteristics of the BMC hedge dataset.

### 3.1.4 Evaluation on scientific texts from a different source

Following the annotation standards of Medlock and Briscoe (Medlock and Briscoe, 2007), we manually annotated 4 full articles downloaded from the

We assumed that these might suggest a speculative assertion.

BMC Bioinformatics website to evaluate our final model on documents from an external source. The chief characteristics of this dataset (which is available at<sup>7</sup>) is shown in Table 1. Surprisingly, the model learnt on FlyBase articles seemed to generalise to these texts only to a limited extent. Our hedge classifier model yielded a  $BEP(spec) = 75.88\%$  and  $F_{\beta=1}(spec) = 74.93\%$  (mainly due to a drop in precision), which is unexpectedly low compared to the previous results.

Analysis of errors revealed that some keywords which proved to be very reliable hedge cues in FlyBase articles were also used in non-speculative contexts in the BMC articles. Over 50% (24 out of 47) of our false positive predictions were due to the different use of 2 keywords, *possible* and *likely*. These keywords were many times used in a mathematical context (referring to probabilities) and thus expressed no speculative meaning, while such uses were not represented in the FlyBase articles (otherwise bigram or trigram features could have captured these non-speculative uses).

### 3.1.5 The effect of using 2-3 word-long phrases as hedge cues

Our experiments demonstrated that it is indeed a good idea to include longer phrases in the vector space model representation of sentences. One third of the features used by our advanced model were either bigrams or trigrams. About half of these were the kind of phrases that had no unigram components of themselves in the feature set, so these could be regarded as meaningful standalone features. Examples of such speculative markers in the fruit fly dataset were: *results support, these observations, indicate that, not clear, does not appear, ...* The majority of these phrases were found to be reliable enough for our maximum entropy model to predict a speculative class based on that single feature.

Our model using just unigram features achieved a  $BEP(spec)$  score of 78.68% and  $F_{\beta=1}(spec)$  score of 80.23%, which means that using bigram and trigram hedge cues here significantly improved the performance (the difference in  $BEP(spec)$  and  $F_{\beta=1}(spec)$  scores were 5.23% and 4.97%, respectively).

<sup>7</sup><http://www.inf.u-szeged.hu/~szarvas/homepage/hedge.html>

## 3.2 Results for hedge classification in radiology reports

In this section we present results using the above-mentioned methods for the automatic detection of speculative assertions in radiology reports. Here we generated training data by an automated procedure. Since hedge cues cause systems to predict false positive labels, our idea here was to train Maximum Entropy Models for the false positive classifications of our ICD-9-CM coding system using the vector space representation of radiology reports. That is, we classified every sentence that contained a medical term (disease or symptom name) and caused the automated ICD-9 coder<sup>8</sup> to predict a false positive code was treated as a speculative sentence and all the rest were treated as non-speculative sentences.

Here a significant part of the false positive predictions of an ICD-9-CM coding system that did not handle hedging originated from speculative assertions, which led us to expect that we would have the most hedge cues among the top ranked keywords which implied false positive labels.

Taking the above points into account, we used the training set of the publicly available ICD-9-CM dataset to build our model and then evaluated each single token by this model to measure their predictivity for a false positive code. Not surprisingly, some of the best hedge cues appeared among the highest ranked features, while some did not (they did not occur frequently enough in the training data to be captured by statistical methods).

For this task, we set the initial  $P(spec|x)$  threshold for filtering to 0.7 since the dataset was generated by a different process and we expected hedge cues to have lower class-conditional probabilities without the effect of the probabilistic data acquisition method that had been applied for scientific texts. Using all 167 terms as keywords that had  $P(spec|x) > 0.7$  resulted in a hedge classifier with an  $F_{\beta=1}(spec)$  score of 64.04%

After the feature selection process 54 keywords were retained. This 54-keyword maxent classifier got an  $F_{\beta=1}(spec)$  score of 79.73%. Plugging this model (without manual filtering) into the ICD-9 coding system as a hedge module, the ICD-9 coder

<sup>8</sup>Here the ICD-9 coding system did not handle the hedging task.

yielded an F measure of 88.64%, which is much better than one without a hedge module (79.7%).

Our experiments revealed that in radiology reports, which mainly concentrate on listing the identified diseases and symptoms (facts) and the physician’s impressions (speculative parts), detecting hedge instances can be performed accurately using unigram features. All bi- and trigrams retained by our feature selection process had unigram equivalents that were eliminated due to the noise present in the automatically generated training data.

We manually examined all keywords that had a  $P(spec) > 0.5$  given as a standalone instance for our maxent model, and constructed a dictionary of hedge cues from the promising candidates. Here we judged 34 out of 54 candidates to be potentially useful for hedging. Using these 34 keywords we got an  $F_{\beta=1}(spec)$  performance of 81.96% due to the improved precision score.

Extending the dictionary with the keywords we gathered from the fruit fly dataset increased the  $F_{\beta=1}(spec)$  score to 82.07% with only one out-domain keyword accepted by the maxent classifier.

	Biomedical papers		Medical reports
	$BEP(spec)$	$F_{\beta=1}(spec)$	$F_{\beta=1}(spec)$
Baseline 1	60.00	–	48.99
Baseline 2	76.30	–	–
All features	76.05	73.61	64.04
Feature selection	78.68	78.09	79.73
Manual feat. sel.	82.02	80.88	81.96
Outer dictionary	85.29	85.08	82.07

Table 2: Summary of results.

## 4 Conclusions

The overall results of our study are summarised in a concise way in Table 2. We list  $BEP(spec)$  and  $F_{\beta=1}(spec)$  values for the scientific text dataset, and  $F_{\beta=1}(spec)$  for the clinical free text dataset. Baseline 1 denotes the substring matching system of Light et al. (Light et al., 2004) and Baseline 2 denotes the system of Medlock and Briscoe (Medlock and Briscoe, 2007). For clinical free texts, Baseline 1 is an out-domain model since the keywords were

collected for scientific texts by (Light et al., 2004). The third row corresponds to a model using all keywords  $P(spec|x)$  above the threshold and the fourth row a model after automatic noise filtering, while the fifth row shows the performance after the manual filtering of automatically selected keywords. The last row shows the benefit gained by adding reliable keywords from an external hedge keyword dictionary.

Our results presented above confirm our hypothesis that speculative language plays an important role in the biomedical domain, and it should be handled in various NLP applications. We experimentally compared the general features of this task in texts from two different domains, namely medical free texts (radiology reports), and scientific articles on the fruit fly from FlyBase.

The radiology reports had mainly unambiguous single-term hedge cues. On the other hand, it proved to be useful to consider bi- and trigrams as hedge cues in scientific texts. This, and the fact that many hedge cues were found to be ambiguous (they appeared in both speculative and non-speculative assertions) can be attributed to the literary style of the articles. Next, as the learnt maximum entropy models show, the hedge classification task reduces to a lookup for single keywords or phrases and to the evaluation of the text based on the most relevant cue alone. Removing those features that were insufficient to classify an instance as a hedge individually did not produce any difference in the  $F_{\beta=1}(spec)$  scores. This latter fact justified a view of ours, namely that during the construction of a statistical hedge detection module for a given application the main issue is to find the task-specific keywords.

Our findings based on the two datasets employed show that automatic or weakly supervised data acquisition, combined with automatic and manual feature selection to eliminate the skewed nature of the data obtained, is a good way of building hedge classifier modules with an acceptable performance.

The analysis of errors indicate that more complex features like dependency structure and clausal phrase information could only help in allocating the scope of hedge cues detected in a sentence, not the detection of any itself. Our finding that token unigram features are capable of solving the task accurately agrees with the the results of previous works on hedge classification ((Light et al., 2004), (Med-

lock and Briscoe, 2007)), and we argue that 2-3 word-long phrases also play an important role as hedge cues and as non-speculative uses of an otherwise speculative keyword as well (i.e. to resolve an ambiguity). In contrast to the findings of Wiebe et al. ((Wiebe et al., 2004)), who addressed the broader task of subjectivity learning and found that the density of other potentially subjective cues in the context benefits classification accuracy, we observed that the co-occurrence of speculative cues in a sentence does not help in classifying a term as speculative or not. Realising that our learnt models never predicted speculative labels based on the presence of two or more individually weak cues and discarding such terms that were not reliable enough to predict a speculative label (using that term alone as a single feature) slightly improved performance, we came to the conclusion that even though speculative keywords tend to cooccur, and two keywords are present in many sentences; hedge cues have a speculative meaning (or not) on their own without the other term having much impact on this.

The main issue thus lies in the selection of keywords, for which we proposed a procedure that is capable of reducing the number of candidates to an acceptable level for human evaluation – even in data collected automatically and thus having some undesirable properties.

The worse results on biomedical scientific papers from a different source also corroborates our finding that hedge cues can be highly ambiguous. In our experiments two keywords that are practically never used in a non-speculative context in the FlyBase articles we used for training were responsible for 50% of false positives in BMC texts since they were used in a different meaning. In our case, the keywords *possible* and *likely* are apparently always used as speculative terms in the FlyBase articles used, while the articles from BMC Bioinformatics frequently used such cliché phrases as *all possible combinations* or *less likely / more likely ...* (referring to probabilities shown in the figures). This shows that the portability of hedge classifiers is limited, and cannot really be done without the examination of the specific features of target texts or a more heterogeneous corpus is required for training. The construction of hedge classifiers for each separate target application in a weakly supervised way seems

feasible though. Collecting bi- and trigrams which cover non-speculative usages of otherwise common hedge cues is a promising solution for addressing the false positives in hedge classifiers and for improving the portability of hedge modules.

#### 4.1 Resolving the scope of hedge keywords

In this paper we focused on the recognition of hedge cues in texts. Another important issue would be to determine the scope of hedge cues in order to locate uncertain sentence parts. This can be solved effectively using a parser adapted for biomedical papers. We manually evaluated the parse trees generated by (Miyao and Tsujii, 2005) and came to the conclusion that for each keyword it is possible to define the scope of the keyword using subtrees linked to the keyword in the predicate-argument syntactic structure or by the immediate subsequent phrase (e.g. prepositional phrase). Naturally, parse errors result in (slightly) mislocated scopes but we had the general impression that state-of-the-art parsers could be used efficiently for this issue. On the other hand, this approach requires a human expert to define the scope for each keyword separately using the predicate-argument relations, or to determine keywords that act similarly and their scope can be located with the same rules. Another possibility is simply to define the scope to be each token up to the end of the sentence (and optionally to the previous punctuation mark). The latter solution has been implemented by us and works accurately for clinical free texts. This simple algorithm is similar to NegEx (Chapman et al., 2001) as we use a list of phrases and their context, but we look for punctuation marks to determine the scopes of keywords instead of applying a fixed window size.

#### Acknowledgments

This work was supported in part by the NKTH grant of Jedlik Ányos R&D Programme 2007 of the Hungarian government (codename TUDORKA7). The author wishes to thank the anonymous reviewers for valuable comments and Veronika Vincze for valuable comments in linguistic issues and for help with the annotation work.

## References

- Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.
- Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of Biomedical Informatics*, 5:301–310.
- Ken Hyland. 1994. Hedging in academic writing and eap textbooks. *English for Specific Purposes*, 13(3):239–256.
- Marc Light, Xin Ying Qiu, and Padmini Srinivasan. 2004. The language of bioscience: Facts, speculations, and statements in between. In Lynette Hirschman and James Pustejovsky, editors, *HLT-NAACL 2004 Workshop: BioLINK 2004, Linking Biological Literature, Ontologies and Databases*, pages 17–24, Boston, Massachusetts, USA, May 6. Association for Computational Linguistics.
- Ben Medlock and Ted Briscoe. 2007. Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 992–999, Prague, Czech Republic, June. Association for Computational Linguistics.
- Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 83–90, Ann Arbor, Michigan, June. Association for Computational Linguistics.
- Marie A. Moio. 2006. *A Guide to Health Insurance Billing*. Thomson Delmar Learning.
- John P. Pestian, Chris Brew, Pawel Matykiewicz, DJ Hovermale, Neil Johnson, K. Bretonnel Cohen, and Wlodzislaw Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Biological, translational, and clinical language processing*, pages 97–104, Prague, Czech Republic, June. Association for Computational Linguistics.
- Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Computational Natural Language Learning Conference*, pages 25–32, Edmonton, Canada, May-June. Association for Computational Linguistics.
- James G. Shanahan, Yan Qu, and Janyce Wiebe. 2005. *Computing Attitude and Affect in Text: Theory and Applications (The Information Retrieval Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- Janyce Wiebe, Theresa Wilson, Rebecca F. Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational Linguistics*, 30(3):277–308.

# When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging

**Alina Andreevskaia**

Concordia University  
Montreal, Quebec

andreev@cs.concordia.ca

**Sabine Bergler**

Concordia University  
Montreal, Canada

bergler@cs.concordia.ca

## Abstract

This study presents a novel approach to the problem of system portability across different domains: a sentiment annotation system that integrates a corpus-based classifier trained on a small set of annotated in-domain data and a lexicon-based system trained on WordNet. The paper explores the challenges of system portability across domains and text genres (movie reviews, news, blogs, and product reviews), highlights the factors affecting system performance on out-of-domain and small-set in-domain data, and presents a new system consisting of the ensemble of two classifiers with precision-based vote weighting, that provides significant gains in accuracy and recall over the corpus-based classifier and the lexicon-based system taken individually.

## 1 Introduction

One of the emerging directions in NLP is the development of machine learning methods that perform well not only on the domain on which they were trained, but also on other domains, for which training data is not available or is not sufficient to ensure adequate machine learning. Many applications require reliable processing of heterogeneous corpora, such as the World Wide Web, where the diversity of genres and domains present in the Internet limits the feasibility of in-domain training. In this paper, sentiment annotation is defined as the assignment of positive, negative or neutral sentiment values to texts, sentences, and other linguistic units. Recent experiments assessing system portability across different domains, conducted by Aue

and Gamon (2005), demonstrated that sentiment annotation classifiers trained in one domain do not perform well on other domains. A number of methods has been proposed in order to overcome this system portability limitation by using out-of-domain data, unlabelled in-domain corpora or a combination of in-domain and out-of-domain examples (Aue and Gamon, 2005; Bai et al., 2005; Drezde et al., 2007; Tan et al., 2007).

In this paper, we present a novel approach to the problem of system portability across different domains by developing a sentiment annotation system that integrates a corpus-based classifier with a lexicon-based system trained on WordNet. By adopting this approach, we sought to develop a system that relies on both general and domain-specific knowledge, as humans do when analyzing a text. The information contained in lexicographical sources, such as WordNet, reflects a lay person's general knowledge about the world, while domain-specific knowledge can be acquired through classifier training on a small set of in-domain data.

The first part of this paper reviews the extant literature on domain adaptation in sentiment analysis and highlights promising directions for research. The second part establishes a baseline for system evaluation by drawing comparisons of system performance across four different domains/genres - movie reviews, news, blogs, and product reviews. The final, third part of the paper presents our system, composed of an ensemble of two classifiers - one trained on WordNet glosses and synsets and the other trained on a small in-domain training set.

## 2 Domain Adaptation in Sentiment Research

Most text-level sentiment classifiers use standard machine learning techniques to learn and select features from labeled corpora. Such approaches work well in situations where large labeled corpora are available for training and validation (e.g., movie reviews), but they do not perform well when training data is scarce or when it comes from a different domain (Aue and Gamon, 2005; Read, 2005), topic (Read, 2005) or time period (Read, 2005). There are two alternatives to supervised machine learning that can be used to get around this problem: on the one hand, general lists of sentiment clues/features can be acquired from domain-independent sources such as dictionaries or the Internet, on the other hand, unsupervised and weakly-supervised approaches can be used to take advantage of a small number of annotated in-domain examples and/or of unlabelled in-domain data.

The first approach, using general word lists automatically acquired from the Internet or from dictionaries, outperforms corpus-based classifiers when such classifiers use out-of-domain training data or when the training corpus is not sufficiently large to accumulate the necessary feature frequency information. But such general word lists were shown to perform worse than statistical models built on sufficiently large in-domain training sets of movie reviews (Pang et al., 2002). On other domains, such as product reviews, the performance of systems that use general word lists is comparable to the performance of supervised machine learning approaches (Gamon and Aue, 2005).

The recognition of major performance deficiencies of supervised machine learning methods with insufficient or out-of-domain training brought about an increased interest in unsupervised and weakly-supervised approaches to feature learning. For instance, Aue and Gamon (2005) proposed training on a small number of labeled examples and large quantities of unlabelled in-domain data. This system performed well even when compared to systems trained on a large set of in-domain examples: on feedback messages from a web survey on knowledge bases, Aue and Gamon report 73.86% accuracy using unlabelled data compared to 77.34% for

in-domain and 72.39% for the best out-of-domain training on a large training set.

Drezde et al. (2007) applied structural correspondence learning (Drezde et al., 2007) to the task of domain adaptation for sentiment classification of product reviews. They showed that, depending on the domain, a small number (e.g., 50) of labeled examples allows to adapt the model learned on another corpus to a new domain. However, they note that the success of such adaptation and the number of necessary in-domain examples depends on the similarity between the original domain and the new one. Similarly, Tan et al. (2007) suggested to combine out-of-domain labeled examples with unlabelled ones from the target domain in order to solve the domain-transfer problem. They applied an out-of-domain-trained SVM classifier to label examples from the target domain and then retrained the classifier using these new examples. In order to maximize the utility of the examples from the target domain, these examples were selected using Similarity Ranking and Relative Similarity Ranking algorithms (Tan et al., 2007). Depending on the similarity between domains, this method brought up to 15% gain compared to the baseline SVM.

Overall, the development of semi-supervised approaches to sentiment tagging is a promising direction of the research in this area but so far, based on reported results, the performance of such methods is inferior to the supervised approaches with in-domain training and to the methods that use general word lists. It also strongly depends on the similarity between the domains as has been shown by (Drezde et al., 2007; Tan et al., 2007).

## 3 Factors Affecting System Performance

The comparison of system performance across different domains involves a number of factors that can significantly affect system performance – from training set size to level of analysis (sentence or entire document), document domain/genre and many other factors. In this section we present a series of experiments conducted to assess the effects of different external factors (i.e., factors unrelated to the merits of the system itself) on system performance in order to establish the baseline for performance comparisons across different domains/genres.

### 3.1 Level of Analysis

Research on sentiment annotation is usually conducted at the text (Aue and Gamon, 2005; Pang et al., 2002; Pang and Lee, 2004; Riloff et al., 2006; Turney, 2002; Turney and Littman, 2003) or at the sentence levels (Gamon and Aue, 2005; Hu and Liu, 2004; Kim and Hovy, 2005; Riloff et al., 2006). It should be noted that each of these levels presents different challenges for sentiment annotation. For example, it has been observed that texts often contain multiple opinions on different topics (Turney, 2002; Wiebe et al., 2001), which makes assignment of the overall sentiment to the whole document problematic. On the other hand, each individual sentence contains a limited number of sentiment clues, which often negatively affects the accuracy and recall if that single sentiment clue encountered in the sentence was not learned by the system.

Since the comparison of sentiment annotation system performance on texts and on sentences has not been attempted to date, we also sought to close this gap in the literature by conducting the first set of our comparative experiments on data sets of 2,002 movie review texts and 10,662 movie review snippets (5331 with positive and 5331 with negative sentiment) provided by Bo Pang (<http://www.cs.cornell.edu/People/pabo/movie-review-data/>).

### 3.2 Domain Effects

The second set of our experiments explores system performance on different domains at sentence level. For this we used four different data sets of sentences annotated with sentiment tags:

- A set of movie review snippets (further: movie) from (Pang and Lee, 2005). This dataset of 10,662 snippets was collected automatically from [www.rottentomatoes.com](http://www.rottentomatoes.com) website. All sentences in reviews marked “rotten” were considered negative and snippets from “fresh” reviews were deemed positive. In order to make the results obtained on this dataset comparable to other domains, a randomly selected subset of 1066 snippets was used in the experiments.
- A balanced corpus of 800 manually annotated sentences extracted from 83 newspaper texts

(further, news). The full set of sentences was annotated by one judge. 200 sentences from this corpus (100 positive and 100 negative) were also randomly selected from the corpus for an inter-annotator agreement study and were manually annotated by two independent annotators. The pairwise agreement between annotators was calculated as the percent of same tags divided by the number of sentences with this tag in the gold standard. The pair-wise agreement between the three annotators ranged from 92.5 to 95.9% ( $\kappa=0.74$  and 0.75 respectively) on positive vs. negative tags.

- A set of sentences taken from personal weblogs (further, blogs) posted on LiveJournal (<http://www.livejournal.com>) and on <http://www.cyberjournalist.com>. This corpus is composed of 800 sentences (400 sentences with positive and 400 sentences with negative sentiment). In order to establish the inter-annotator agreement, two independent judges were asked to annotate 200 sentences from this corpus. The agreement between the two annotators on positive vs. negative tags reached 99% ( $\kappa=0.97$ ).
- A set of 1200 product review (PR) sentences extracted from the annotated corpus made available by Bing Liu (Hu and Liu, 2004) (<http://www.cs.uic.edu/liub/FBS/FBS.html>).

The data set sizes are summarized in Table 1.

	Movies	News	Blogs	PR
Text level	2002 texts	n/a	n/a	n/a
Sentence level	10662 snippets	800 sent.	800 sent.	1200 sent.

Table 1: Datasets

### 3.3 Establishing a Baseline for a Corpus-based System (CBS)

Supervised statistical methods have been very successful in sentiment tagging of texts: on movie review texts they reach accuracies of 85-90% (Aue and Gamon, 2005; Pang and Lee, 2004). These methods perform particularly well when a large volume of labeled data from the same domain as the



test set is available for training (Aue and Gamon, 2005). For this reason, most of the research on sentiment tagging using statistical classifiers was limited to product and movie reviews, where review authors usually indicate their sentiment in a form of a standardized score that accompanies the texts of their reviews.

The lack of sufficient data for training appears to be the main reason for the virtual absence of experiments with statistical classifiers in sentiment tagging at the sentence level. To our knowledge, the only work that describes the application of statistical classifiers (SVM) to sentence-level sentiment classification is (Gamon and Aue, 2005)<sup>1</sup>. The average performance of the system on ternary classification (positive, negative, and neutral) was between 0.50 and 0.52 for both average precision and recall. The results reported by (Riloff et al., 2006) for binary classification of sentences in a related domain of *subjectivity* tagging (i.e., the separation of sentiment-laden from neutral sentences) suggest that statistical classifiers can perform well on this task: the authors have reached 74.9% accuracy on the MPQA corpus (Riloff et al., 2006).

In order to explore the performance of different approaches in sentiment annotation at the text and sentence levels, we used a basic Naïve Bayes classifier. It has been shown that both Naïve Bayes and SVMs perform with similar accuracy on different sentiment tagging tasks (Pang and Lee, 2004). These observations were confirmed with our own experiments with SVMs and Naïve Bayes (Table 3). We used the Weka package (<http://www.cs.waikato.ac.nz/ml/weka/>) with default settings.

In the sections that follow, we describe a set of comparative experiments with SVMs and Naïve Bayes classifiers (1) on texts and sentences and (2) on four different domains (movie reviews, news, blogs, and product reviews). System runs with unigrams, bigrams, and trigrams as features and with different training set sizes are presented.

<sup>1</sup>Recently, a similar task has been addressed by the Affective Text Task at SemEval-1 where even shorter units – headlines – were classified into positive, negative and neutral categories using a variety of techniques (Strapparava and Mihalcea, 2007).

## 4 Experiments

### 4.1 System Performance on Texts vs. Sentences

The experiments comparing in-domain trained system performance on texts vs. sentences were conducted on 2,002 movie review texts and on 10,662 movie review snippets. The results with 10-fold cross-validation are reported in Table 2<sup>2</sup>.

	Trained on Texts		Trained on Sent.	
	Tested on Texts	Tested on Sent.	Tested on Texts	Tested on Sent.
1gram	<b>81.1</b>	69.0	66.8	<b>77.4</b>
2gram	<b>83.7</b>	68.6	71.2	<b>73.9</b>
3gram	<b>82.5</b>	64.1	70.0	<b>65.4</b>

Table 2: Accuracy of Naïve Bayes on movie reviews.

Consistent with findings in the literature (Cui et al., 2006; Dave et al., 2003; Gamon and Aue, 2005), on the large corpus of movie review texts, the in-domain-trained system based solely on unigrams had lower accuracy than the similar system trained on bigrams. But the trigrams fared slightly worse than bigrams. On sentences, however, we have observed an inverse pattern: unigrams performed better than bigrams and trigrams. These results highlight a special property of sentence-level annotation: *greater sensitivity to sparseness* of the model: On texts, classifier error on one particular sentiment marker is often compensated by a number of correctly identified other sentiment clues. Since sentences usually contain a much smaller number of sentiment clues than texts, sentence-level annotation more readily yields errors when a single sentiment clue is incorrectly identified or missed by the system. Due to lower frequency of higher-order n-grams (as opposed to unigrams), higher-order n-gram language models are more sparse, which increases the probability of missing a particular sentiment marker in a sentence (Table 3<sup>3</sup>). Very large

<sup>2</sup>All results are statistically significant at  $\alpha = 0.01$  with two exceptions: the difference between trigrams and bigrams for the system trained and tested on texts is statistically significant at  $\alpha=0.1$  and for the system trained on sentences and tested on texts is not statistically significant at  $\alpha = 0.01$ .

<sup>3</sup>The results for movie reviews are lower than those reported in Table 2 since the dataset is 10 times smaller, which results in less accurate classification. The statistical significance of the

training sets are required to overcome this higher n-gram sparseness in sentence-level annotation.

Dataset	Movie	News	Blogs	PRs
Dataset size	1066	800	800	1200
unigrams				
SVM	68.5	61.5	63.85	76.9
NB	60.2	59.5	60.5	74.25
nb features	5410	4544	3615	2832
bigrams				
SVM	59.9	63.2	61.5	75.9
NB	57.0	58.4	59.5	67.8
nb features	16286	14633	15182	12951
trigrams				
SVM	54.3	55.4	52.7	64.4
NB	53.3	57.0	56.0	69.7
nb features	20837	18738	19847	19132

Table 3: Accuracy of unigram, bigram and trigram models across domains.

## 4.2 System Performance on Different Domains

In the second set of experiments we sought to compare system results on sentences using in-domain and out-of-domain training. Table 4 shows that in-domain training, as expected, consistently yields superior accuracy than out-of-domain training across all four datasets: movie reviews (Movies), news, blogs, and product reviews (PRs). The numbers for in-domain trained runs are highlighted in bold.

Training Data	Test Data			
	Movies	News	Blogs	PRs
Movies	<b>68.5</b>	55.2	53.2	60.7
News	55.0	<b>61.5</b>	56.25	57.4
Blogs	53.7	49.9	<b>63.85</b>	58.8
PRs	55.8	55.9	56.25	<b>76.9</b>

Table 4: Accuracy of SVM with unigram model

results depends on the genre and size of the n-gram: on product reviews, all results are statistically significant at  $\alpha = 0.025$  level; on movie reviews, the difference between Naïve Bayes and SVM is statistically significant at  $\alpha = 0.01$  but the significance diminishes as the size of the n-gram increases; on news, only bi-grams produce a statistically significant ( $\alpha = 0.01$ ) difference between the two machine learning methods, while on blogs the difference between SVMs and Naïve Bayes is most pronounced when unigrams are used ( $\alpha = 0.025$ ).

It is interesting to note that *on sentences*, regardless of the domain used in system training and regardless of the domain used in system testing, unigrams tend to perform better than higher-order n-grams. This observation suggests that, given the constraints on the size of the available training sets, unigram-based systems may be better suited for sentence-level sentiment annotation.

## 5 Lexicon-Based Approach

The search for a base-learner that can produce greatest synergies with a classifier trained on small-set in-domain data has turned our attention to lexicon-based systems. Since the benefits from combining classifiers that always make similar decisions is minimal, the two (or more) base-learners should complement each other (Alpaydin, 2004). Since a system based on a fairly different learning approach is more likely to produce a different decision under a given set of circumstances, the diversity of approaches integrated in the ensemble of classifiers was expected to have a beneficial effect on the overall system performance.

A lexicon-based approach capitalizes on the fact that dictionaries, such as WordNet (Fellbaum, 1998), contain a comprehensive and domain-independent set of sentiment clues that exist in general English. A system trained on such general data, therefore, should be less sensitive to domain changes. This robustness, however is expected to come at some cost, since some domain-specific sentiment clues may not be covered in the dictionary. Our hypothesis was, therefore, that a lexicon-based system will perform worse than an in-domain trained classifier but possibly better than a classifier trained on out-of domain data.

One of the limitations of general lexicons and dictionaries, such as WordNet (Fellbaum, 1998), as training sets for sentiment tagging systems is that they contain only definitions of individual words and, hence, only unigrams could be effectively learned from dictionary entries. Since the structure of WordNet glosses is fairly different from that of other types of corpora, we developed a system that used the list of human-annotated adjectives from (Hatzivassiloglou and McKeown, 1997) as a seed list and then learned additional unigrams

from WordNet synsets and glosses with up to 88% accuracy, when evaluated against General Inquirer (Stone et al., 1966) (GI) on the intersection of our automatically acquired list with GI. In order to expand the list coverage for our experiments at the text and sentence levels, we then augmented the list by adding to it all the words annotated with “Positiv” or “Negativ” tags in GI, that were not picked up by the system. The resulting list of features contained 11,000 unigrams with the degree of membership in the category of positive or negative sentiment assigned to each of them.

In order to assign the membership score to each word, we did 58 system runs on unique non-intersecting seed lists drawn from manually annotated list of positive and negative adjectives from (Hatzivassiloglou and McKeown, 1997). The 58 runs were then collapsed into a single set of 7,813 unique words. For each word we computed a score by subtracting the total number of runs assigning this word a negative sentiment from the total of the runs that consider it positive. The resulting measure, termed Net Overlap Score (NOS), reflected the number of ties linking a given word with other sentiment-laden words in WordNet, and hence, could be used as a measure of the words’ centrality in the fuzzy category of sentiment. The NOSs were then normalized into the interval from -1 to +1 using a sigmoid fuzzy membership function (Zadeh, 1975)<sup>4</sup>. Only words with fuzzy membership degree not equal to zero were retained in the list. The resulting list contained 10,809 sentiment-bearing words of different parts of speech. The sentiment determination at the sentence and text level was then done by summing up the scores of all identified positive unigrams (NOS>0) and all negative unigrams (NOS<0) (Andreevskaia and Bergler, 2006).

### 5.1 Establishing a Baseline for the Lexicon-Based System (LBS)

The baseline performance of the Lexicon-Based System (LBS) described above is presented in Table 5, along with the performance results of the in-domain- and out-of-domain-trained SVM classifier.

Table 5 confirms the predicted pattern: the LBS performs with lower accuracy than *in-domain-*

	Movies	News	Blogs	PRs
LBS	57.5	62.3	63.3	59.3
SVM in-dom.	68.5	61.5	63.85	76.9
SVM out-of-dom.	55.8	55.9	56.25	60.7

Table 5: System accuracy on best runs on sentences

trained corpus-based classifiers, and with similar or better accuracy than the corpus-based classifiers trained on *out-of-domain* data. Thus, the lexicon-based approach is characterized by a bounded but stable performance when the system is ported across domains. These performance characteristics of corpus-based and lexicon-based approaches prompt further investigation into the possibility to combine the portability of dictionary-trained systems with the accuracy of in-domain trained systems.

## 6 Integrating the Corpus-based and Dictionary-based Approaches

The strategy of integration of two or more systems in a single ensemble of classifiers has been actively used on different tasks within NLP. In sentiment tagging and related areas, Aue and Gamon (2005) demonstrated that combining classifiers can be a valuable tool in domain adaptation for sentiment analysis. In the ensemble of classifiers, they used a combination of nine SVM-based classifiers deployed to learn unigrams, bigrams, and trigrams on three different domains, while the fourth domain was used as an evaluation set. Using then an SVM meta-classifier trained on a small number of target domain examples to combine the nine base classifiers, they obtained a statistically significant improvement on out-of-domain texts from book reviews, knowledge-base feedback, and product support services survey data. No improvement occurred on movie reviews.

Pang and Lee (2004) applied two different classifiers to perform sentiment annotation in two sequential steps: the first classifier separated subjective (sentiment-laden) texts from objective (neutral) ones and then they used the second classifier to classify the subjective texts into positive and negative. Das and Chen (2004) used five classifiers to determine market sentiment on Yahoo! postings. Simple majority vote was applied to make decisions within

<sup>4</sup>With coefficients:  $\alpha=1, \gamma=15$ .

the ensemble of classifiers and achieved accuracy of 62% on ternary in-domain classification.

In this study we describe a system that attempts to combine the portability of a dictionary-trained system (LBS) with the accuracy of an in-domain trained corpus-based system (CBS). The selection of these two classifiers for this system, thus, was theory-based. The section that follows describes the classifier integration and presents the performance results of the system consisting of an ensemble CBS and LBS classifier and a precision-based vote weighting procedure.

### 6.1 The Classifier Integration Procedure and System Evaluation

The comparative analysis of the corpus-based and lexicon-based systems described above revealed that the errors produced by CBS and LBS were to a great extent complementary (i.e., where one classifier makes an error, the other tends to give the correct answer). This provided further justification to the integration of corpus-based and lexicon-based approaches in a single system.

Table 6 below illustrates the complementarity of the performance CBS and LBS classifiers on the positive and negative categories. In this experiment, the corpus-based classifier was trained on 400 annotated product review sentences<sup>5</sup>. The two systems were then evaluated on a test set of another 400 product review sentences. The results reported in Table 6 are statistically significant at  $\alpha = 0.01$ .

	CBS	LBS
Precision positives	<b>89.3%</b>	69.3%
Precision negatives	55.5%	<b>81.5%</b>
Pos/Neg Precision	58.0%	72.1%

Table 6: Base-learners' precision and recall on product reviews on test data.

Table 6 shows that the corpus-based system has a very good precision on those sentences that it classifies as positive but makes a lot of errors on those sentences that it deems negative. At the same time, the lexicon-based system has low precision on positives

<sup>5</sup>The small training set explains relatively low overall performance of the CBS system.

and high precision on negatives<sup>6</sup>. Such complementary distribution of errors produced by the two systems was observed on different data sets from different domains, which suggests that the observed distribution pattern reflects the properties of each of the classifiers, rather than the specifics of the domain/genre.

In order to take advantage of the observed complementarity of the two systems, the following procedure was used. First, a small set of in-domain data was used to train the CBS system. Then both CBS and LBS systems were run separately on the same training set, and for each classifier, the precision measures were calculated separately for those sentences that the classifier considered positive and those it considered negative. The chance-level performance (50%) was then subtracted from the precision figures to ensure that the final weights reflect by how much the classifier's precision exceeds the chance level. The resulting chance-adjusted precision numbers of the two classifiers were then normalized, so that the weights of CBS and LBS classifiers sum up to 100% on positive and to 100% on negative sentences. These weights were then used to adjust the contribution of each classifier to the decision of the ensemble system. The choice of the weight applied to the classifier decision, thus, varied depending on whether the classifier scored a given sentence as positive or as negative. The resulting system was then tested on a separate test set of sentences<sup>7</sup>. The small-set training and evaluation experiments with the system were performed on different domains using 3-fold validation.

The experiments conducted with the Ensemble system were designed to explore system performance under conditions of limited availability of annotated data for classifier training. For this reason, the numbers reported for the corpus-based classifier do not reflect the full potential of machine learning approaches when sufficient in-domain training data is available. Table 7 presents the results of these experiments by domain/genre. The results

<sup>6</sup>These results are consistent with an observation in (Kennedy and Inkpen, 2006), where a lexicon-based system performed with a better precision on negative than on positive texts.

<sup>7</sup>The size of the test set varied in different experiments due to the availability of annotated data for a particular domain.

are statistically significant at  $\alpha = 0.01$ , except the runs on movie reviews where the difference between the LBS and Ensemble classifiers was significant at  $\alpha = 0.05$ .

	LBS	CBS	Ensemble
News	Acc 67.8	53.2	<b>73.3</b>
	F 0.82	0.71	<b>0.85</b>
Movies	Acc 54.5	53.5	<b>62.1</b>
	F 0.73	0.72	<b>0.77</b>
Blogs	Acc 61.2	51.1	<b>70.9</b>
	F 0.78	0.69	<b>0.83</b>
PRs	Acc 59.5	58.9	<b>78.0</b>
	F 0.77	0.75	<b>0.88</b>
Average	Acc 60.7	54.2	71.1
	F 0.77	0.72	0.83

Table 7: Performance of the ensemble classifier

Table 7 shows that the combination of two classifiers into an ensemble using the weighting technique described above leads to consistent improvement in system performance across all domains/genres. In the ensemble system, the average gain in accuracy across the four domains was 16.9% relative to CBS and 10.3% relative to LBS. Moreover, the gain in accuracy and precision was not offset by decreases in recall: the net gain in recall was 7.4% relative to CBS and 13.5% vs. LBS. The ensemble system on average reached 99.1% recall. The F-measure has increased from 0.77 and 0.72 for LBS and CBS classifiers respectively to 0.83 for the whole ensemble system.

## 7 Discussion

The development of domain-independent sentiment determination systems poses a substantial challenge for researchers in NLP and artificial intelligence. The results presented in this study suggest that the integration of two fairly different classifier learning approaches in a single ensemble of classifiers can yield substantial gains in system performance on all measures. The most substantial gains occurred in recall, accuracy, and F-measure.

This study permits to highlight a set of factors that enable substantial performance gains with the ensemble of classifiers approach. Such gains are most likely when (1) the errors made by the clas-

sifiers are complementary, i.e., where one classifier makes an error, the other tends to give the correct answer, (2) the classifier errors are not fully random and occur more often in a certain segment (or category) of classifier results, and (3) there is a way for a system to identify that low-precision segment and reduce the weights of that classifier’s results on that segment accordingly. The two classifiers used in this study – corpus-based and lexicon-based – provided an interesting illustration of potential performance gains associated with these three conditions. The use of precision of classifier results on the positives and negatives proved to be an effective technique for classifier vote weighting within the ensemble.

## 8 Conclusion

This study contributes to the research on sentiment tagging, domain adaptation, and the development of ensembles of classifiers (1) by proposing a novel approach for sentiment determination at sentence level and delineating the conditions under which greatest synergies among combined classifiers can be achieved, (2) by describing a precision-based technique for assigning differential weights to classifier results on different categories identified by the classifier (i.e., categories of positive vs. negative sentences), and (3) by proposing a new method for sentiment annotation in situations where the annotated in-domain data is scarce and insufficient to ensure adequate performance of the corpus-based classifier, which still remains the preferred choice when large volumes of annotated data are available for system training.

Among the most promising directions for future research in the direction laid out in this paper is the deployment of more advanced classifiers and feature selection techniques that can further enhance the performance of the ensemble of classifiers. The precision-based vote weighting technique may prove to be effective also in situations, where more than two classifiers are integrated into a single system. We expect that these more advanced ensemble-of-classifiers systems would inherit the benefits of multiple complementary approaches to sentiment annotation and will be able to achieve better and more stable accuracy on in-domain, as well as on out-of-domain data.

## References

- Ethem Alpaydin. 2004. *Introduction to Machine Learning*. The MIT Press, Cambridge, MA.
- Alina Andreevskaia and Sabine Bergler. 2006. Mining WordNet for a fuzzy sentiment: Sentiment tag extraction from WordNet glosses. In *Proceedings the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, IT.
- Anthony Aue and Michael Gamon. 2005. Customizing sentiment classifiers to new domains: a case study. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, Borovets, BG.
- Xue Bai, Rema Padman, and Edoardo Airoldi. 2005. On learning parsimonious models for extracting consumer opinions. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Washington, DC.
- Hang Cui, Vibhu Mittal, and Mayur Datar. 2006. Comparative experiments on sentiment classification for online product reviews. In *Proceedings of the 21st International Conference on Artificial Intelligence*, Boston, MA.
- Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the Peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of WWW03*, Budapest, HU.
- Mark Drezde, John Blitzer, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, CZ.
- Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.
- Michael Gamon and Anthony Aue. 2005. Automatic identification of sentiment vocabulary: exploiting low association with known sentiment terms. In *Proceedings of the ACL-05 Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, Ann Arbor, US.
- Vasileios Hatzivassiloglou and Kathleen B. McKeown. 1997. Predicting the Semantic Orientation of Adjectives. In *Proceedings of the the 40th Annual Meeting of the Association of Computational Linguistics*.
- Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD-04*, pages 168–177.
- Alistair Kennedy and Diana Inkpen. 2006. Sentiment Classification of Movie Reviews Using Contextual Valence Shifters. *Computational Intelligence*, 22(2):110–125.
- Soo-Min Kim and Eduard Hovy. 2005. Automatic detection of opinion bearing words and sentences. In *Proceedings of the Second International Joint Conference on Natural Language Processing, Companion Volume*, Jeju Island, KR.
- Bo Pang and Lillian Lee. 2004. A sentiment education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*.
- Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, Ann Arbor, US.
- Bo Pang, Lillian Lee, and Shrivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Conference on Empirical Methods in Natural Language Processing*.
- Jonathon Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL-2005 Student Research Workshop*, Ann Arbor, MI.
- Ellen Riloff, Siddharth Patwardhan, and Janyce Wiebe. 2006. Feature subsumption for opinion analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Sydney, AUS.
- P.J. Stone, D.C. Dumphy, M.S. Smith, and D.M. Ogilvie. 1966. *The General Inquirer: a computer approach to content analysis*. M.I.T. studies in comparative politics. M.I.T. Press, Cambridge, MA.
- Carlo Strapparava and Rada Mihalcea. 2007. SemEval-2007 Task 14: Affective Text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, Prague, CZ.
- Songbo Tan, Gaowei Wu, Huifeng Tang, and Zueqi Cheng. 2007. A Novel Scheme for Domain-transfer Problem in the context of Sentiment Analysis. In *Proceedings of CIKM 2007*.
- Peter Turney and Michael Littman. 2003. Measuring praise and criticism: inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21:315–346.
- Peter 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 of Computational Linguistics*.
- Janyce Wiebe, Rebecca Bruce, Matthew Bell, Melanie Martin, and Theresa Wilson. 2001. A corpus study of Evaluative and Speculative Language. In *Proceedings of the 2nd ACL SIGDial Workshop on Discourse and Dialogue*, Aalborg, DK.
- Lotfy A. Zadeh. 1975. Calculus of Fuzzy Restrictions. In L.A. Zadeh et al., editor, *Fuzzy Sets and their Applications to cognitive and decision processes*, pages 1–40. Academic Press Inc., New-York.

# A Generic Sentence Trimmer with CRFs

**Tadashi Nomoto**

National Institute of Japanese Literature  
10-3, Midori Tachikawa  
Tokyo, 190-0014, Japan  
nomoto@acm.org

## Abstract

The paper presents a novel sentence trimmer in Japanese, which combines a non-statistical yet generic tree generation model and Conditional Random Fields (CRFs), to address improving the grammaticality of compression while retaining its relevance. Experiments found that the present approach outperforms in grammaticality and in relevance a dependency-centric approach (Oguro et al., 2000; Morooka et al., 2004; Yamagata et al., 2006; Fukutomi et al., 2007) – the only line of work in prior literature (on Japanese compression) we are aware of that allows replication and permits a direct comparison.

## 1 Introduction

For better or worse, much of prior work on sentence compression (Riezler et al., 2003; McDonald, 2006; Turner and Charniak, 2005) turned to a single corpus developed by Knight and Marcu (2002) (K&M, henceforth) for evaluating their approaches.

The K&M corpus is a moderately sized corpus consisting of 1,087 pairs of sentence and compression, which account for about 2% of a Ziff-Davis collection from which it was derived. Despite its limited scale, prior work in sentence compression relied heavily on this particular corpus for establishing results (Turner and Charniak, 2005; McDonald, 2006; Clarke and Lapata, 2006; Galley and McKeown, 2007). It was not until recently that researchers started to turn attention to an alternative approach which does not require supervised data (Turner and Charniak, 2005).

Our approach is broadly in line with prior work (Jing, 2000; Dorr et al., 2003; Riezler et al., 2003; Clarke and Lapata, 2006), in that we make use of some form of syntactic knowledge to constrain compressions we generate. What sets this work apart from them, however, is a novel use we make of Conditional Random Fields (CRFs) to select among possible compressions (Lafferty et al., 2001; Sutton and McCallum, 2006). An obvious benefit of using CRFs for sentence compression is that the model provides a general (and principled) probabilistic framework which permits information from various sources to be integrated towards compressing sentence, a property K&M do not share.

Nonetheless, there is some cost that comes with the straightforward use of CRFs as a discriminative classifier in sentence compression; its outputs are often ungrammatical and it allows no control over the length of compression they generates (Nomoto, 2007). We tackle the issues by harnessing CRFs with what we might call dependency truncation, whose goal is to restrict CRFs to working with candidates that conform to the grammar.

Thus, unlike McDonald (2006), Clarke and Lapata (2006) and Cohn and Lapata (2007), we do not insist on finding a globally optimal solution in the space of  $2^n$  possible compressions for an  $n$  word long sentence. Rather we insist on finding a most plausible compression among those that are explicitly warranted by the grammar.

Later in the paper, we will introduce an approach called the ‘Dependency Path Model’ (DPM) from the previous literature (Section 4), which purports to provide a robust framework for sentence compression.

sion in Japanese. We will look at how the present approach compares with that of DPM in Section 6.

## 2 A Sentence Trimmer with CRFs

Our idea on how to make CRFs comply with grammar is quite simple: we focus on only those label sequences that are associated with grammatically correct compressions, by making CRFs look at only those that comply with some grammatical constraints  $G$ , and ignore others, regardless of how probable they are.<sup>1</sup> But how do we find compressions that are grammatical? To address the issue, rather than resort to statistical generation models as in the previous literature (Cohn and Lapata, 2007; Galley and McKeown, 2007), we pursue a particular rule-based approach we call a ‘dependency truncation,’ which as we will see, gives us a greater control over the form that compression takes.

Let us denote a set of label assignments for  $S$  that satisfy constraints, by  $G(S)$ .<sup>2</sup> We seek to solve the following,

$$\mathbf{y}^* = \arg \max_{\mathbf{y} \in G(S)} p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}). \quad (2)$$

There would be a number of ways to go about the problem. In the context of sentence compression, a linear programming based approach such as Clarke and Lapata (2006) is certainly one that deserves consideration. In this paper, however, we will explore a much simpler approach which does not require as involved formulation as Clarke and Lapata (2006) do.

We approach the problem *extensionally*, i.e., through generating sentences that are grammatical, or that conform to whatever constraints there are.

<sup>1</sup>Assume as usual that CRFs take the form,

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &\propto \\ &\exp\left(\sum_{k,j} \lambda_j f_j(y_k, y_{k-1}, \mathbf{x}) + \sum_i \mu_i g_i(x_k, y_k, \mathbf{x})\right) \\ &= \exp[\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})] \end{aligned} \quad (1)$$

$f_j$  and  $g_i$  are ‘features’ associated with edges and vertices, respectively, and  $k \in \mathcal{C}$ , where  $\mathcal{C}$  denotes a set of cliques in CRFs.  $\lambda_j$  and  $\mu_i$  are the weights for corresponding features.  $\mathbf{w}$  and  $\mathbf{f}$  are vector representations of weights and features, respectively (Tasker, 2004).

<sup>2</sup>Note that a sentence compression can be represented as an array of binary labels, one of them marking words to be retained in compression and the other those to be dropped.

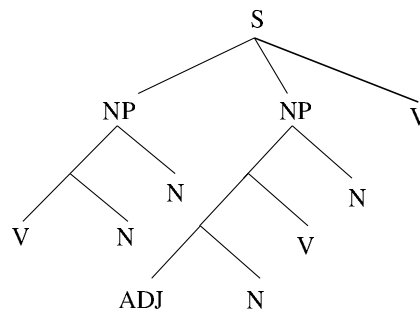


Figure 1: Syntactic structure in Japanese

Consider the following.

- (3) *Mushoku-no John -ga takai kuruma*  
 unemployed John SBJ expensive car  
*-wo kat-ta.*  
 ACC buy PAST  
 ‘John, who is unemployed, bought an expensive car.’

whose grammatically legitimate compressions would include:

- (4) (a) *John -ga takai kuruma -wo kat-ta.*  
 ‘John bought an expensive car.’  
 (b) *John -ga kuruma -wo kat-ta.*  
 ‘John bought a car.’  
 (c) *Mushoku-no John -ga kuruma -wo kat-ta.*  
 ‘John, who is unemployed, bought a car.’  
 (d) *John -ga kat-ta.*  
 ‘John bought.’  
 (e) *Mushoku-no John -ga kat-ta.*  
 ‘John, who is unemployed, bought.’  
 (f) *Takai kuruma-wo kat-ta.*  
 ‘Bought an expensive car.’  
 (g) *Kuruma-wo kat-ta.*  
 ‘Bought a car.’  
 (h) *Kat-ta.*  
 ‘Bought.’

This would give us  $G(S)=\{a, b, c, d, e, f, g, h\}$ , for the input 3. Whatever choice we make for compression among candidates in  $G(S)$ , should be grammatical, since they all are. One linguistic feature



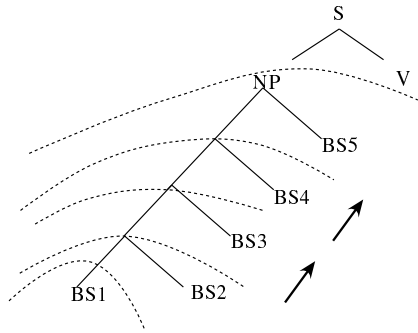


Figure 2: Compressing an NP chunk

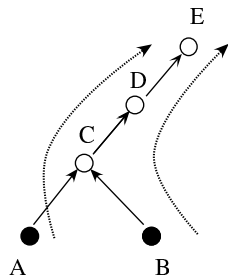


Figure 3: Trimming TDPs

of the Japanese language we need to take into account when generating compressions, is that the sentence, which is free of word order and verb-final, typically takes a left-branching structure as in Figure 1, consisting of an array of morphological units called *bunsetsu* (BS, henceforth). A BS, which we might regard as an inflected form (case marked in the case of nouns) of verb, adjective, and noun, could involve one or more independent linguistic elements such as noun, case particle, but acts as a morphological atom, in that it cannot be torn apart, or partially deleted, without compromising the grammaticality.<sup>3</sup>

Noting that a Japanese sentence typically consists of a sequence of case marked NPs and adjuncts, followed by a main verb at the end (or what would be called ‘matrix verb’ in linguistics), we seek to compress each of the major chunks in the sentence, leaving untouched the matrix verb, as its removal often leaves the sentence unintelligible. In particular, starting with the leftmost BS in a major constituent,

<sup>3</sup>Example 3 could be broken into BSs: / *Mushuku -no / John -ga / takai / kuruma -wo / kat-ta /*.

we work up the tree by pruning BSs on our way up, which in general gives rise to grammatically legitimate compressions of various lengths (Figure 2).

More specifically, we take the following steps to construct  $G(S)$ . Let  $S = ABCDE$ . Assume that it has a dependency structure as in Figure 3. We begin by locating terminal nodes, i.e., those which have no incoming edges, depicted as filled circles in Figure 3, and find a dependency (singly linked) path from each terminal node to the root, or a node labeled ‘E’ here, which would give us two paths  $p_1 = A-C-D-E$  and  $p_2 = B-C-D-E$  (call them *terminating dependency paths*, or TDPs). Now create a set  $\mathcal{T}$  of all trimmings, or suffixes of each TDP, including an empty string:

$$\begin{aligned} \mathcal{T}(p_1) &= \{ \langle A C D E \rangle, \langle C D E \rangle, \langle D E \rangle, \langle E \rangle, \langle \rangle \} \\ \mathcal{T}(p_2) &= \{ \langle B C D E \rangle, \langle C D E \rangle, \langle D E \rangle, \langle E \rangle, \langle \rangle \} \end{aligned}$$

Then we merge subpaths from the two sets in every possible way, i.e., for any two subpaths  $t_1 \in \mathcal{T}(p_1)$  and  $t_2 \in \mathcal{T}(p_2)$ , we take a union over nodes in  $t_1$  and  $t_2$ ; Figure 4 shows how this might be done. We remove duplicates if any. This would give us  $G(S) = \{ \{ A B C D E \}, \{ A C D E \}, \{ B C D E \}, \{ C D E \}, \{ D E \}, \{ E \}, \{ \} \}$ , a set of compressions over  $S$  based on TDPs.

What is interesting about the idea is that creating  $G(S)$  does not involve much of anything that is specific to a given language. Indeed this could be done on English as well. Take for instance a sentence at the top of Table 1, which is a slightly modified lead sentence from an article in the *New York Times*. Assume that we have a relevant dependency structure as shown in Figure 5, where we have three TDPs, i.e., one with *southern*, one with *British* and one with *lethal*. Then  $G(S)$  would include those listed in Table 1. A major difference from Japanese lies in the direction in which a tree is branching out: right versus left.<sup>4</sup>

Having said this, we need to address some language specific constraints: in Japanese, for instance, we should keep a topic marked NP in compression as its removal often leads to a decreased readability; and also it is grammatically wrong to start any compressed segment with sentence nominalizers such as

<sup>4</sup>We stand in a marked contrast to previous ‘grafting’ approaches which more or less rely on an ad-hoc collection of transformation rules to generate candidates (Riezler et al., 2003).

Table 1: Hedge-clipping English

An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks on British troops in southern Iraq

---

An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks on British troops in Iraq

An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks on British troops

An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks on troops

An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks

An official was quoted yesterday as accusing Iran of supplying explosive technology used in attacks

An official was quoted yesterday as accusing Iran of supplying explosive technology

An official was quoted yesterday as accusing Iran of supplying technology

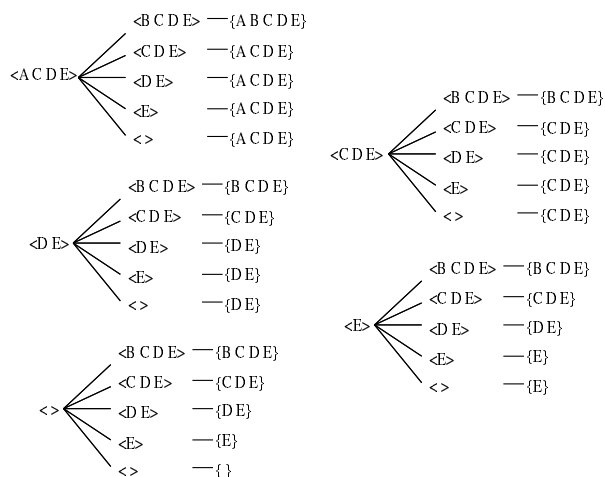


Figure 4: Combining TDP suffixes

*-koto* and *-no*. In English, we should keep a preposition from being left dangling, as in *An official was quoted yesterday as accusing Iran of supplying technology used in*. In any case, we need some extra rules on  $G(S)$  to take care of language specific issues (cf. Vandeghinste and Pan (2004) for English). An important point about the dependency truncation is that for most of the time, a compression it generates comes out reasonably grammatical, so the number of ‘extras’ should be small.

Finally, in order for CRFs to work with the compressions, we need to translate them into a sequence of binary labels, which involves labeling an element token, *bunsetsu* or a word, with some label, e.g., 0 for ‘remove’ and 1 for ‘retain,’ as in Figure 6.

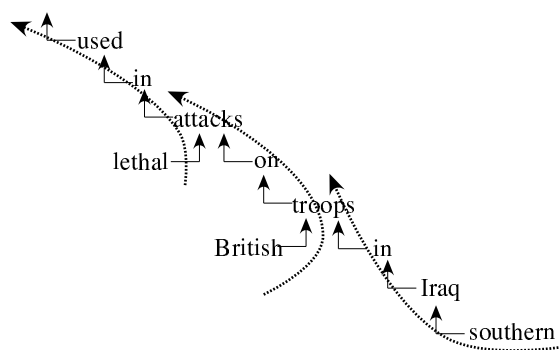


Figure 5: An English dependency structure and TDPs

Consider following compressions  $y_1$  to  $y_4$  for  $x = \beta_1\beta_2\beta_3\beta_4\beta_5\beta_6$ .  $\beta_i$  denotes a *bunsetsu* (BS). ‘0’ marks a BS to be removed and ‘1’ that to be retained.

	$\beta_1$	$\beta_2$	$\beta_3$	$\beta_4$	$\beta_5$	$\beta_6$
$y_1$	0	1	1	1	1	1
$y_2$	0	0	1	1	1	1
$y_3$	0	0	0	0	0	1
$y_4$	0	0	1	0	0	0

Assume that  $G(S) = \{y_1, y_2, y_3\}$ . Because  $y_4$  is not part of  $G(S)$ , it is not considered a candidate for a compression for  $y$ , even if its likelihood may exceed those of others in  $G(S)$ . We note that the approach here does not rely on so much of CRFs as a discriminative classifier as CRFs as a strategy for ranking among a limited set of label sequences which correspond to syntactically plausible simplifications of input sentence.

Furthermore, we could dictate the length of compression by putting an additional constraint on out-

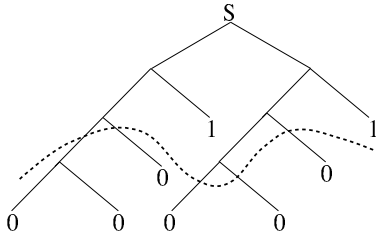


Figure 6: Compression in binary representation.

put, as in:

$$\mathbf{y}^* = \arg \max_{\mathbf{y} \in G'(S)} p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}), \quad (5)$$

where  $G'(S) = \{\mathbf{y} : \mathbf{y} \in G(S), R(\mathbf{y}, \mathbf{x}) = r\}$ .  $R(\mathbf{y}, \mathbf{x})$  denotes a compression rate  $r$  for which  $\mathbf{y}$  is desired, where  $r = \frac{\# \text{ of } 1 \text{ in } \mathbf{y}}{\text{length of } \mathbf{x}}$ . The constraint forces the trimmer to look for the best solution among candidates that satisfy the constraint, ignoring those that do not.<sup>5</sup>

Another point to note is that  $G(S)$  is finite and relatively small – it was found, for our domain,  $G(S)$  usually runs somewhere between a few hundred and ten thousand in length – so in practice it suffices that we visit each compression in  $G(S)$ , and select one that gives the maximum value for the objective function. We will have more to say about the size of the search space in Section 6.

### 3 Features in CRFs

We use an array of features in CRFs which are either derived or borrowed from the taxonomy that a Japanese tokenizer called JUMAN and KNP,<sup>6</sup> a Japanese dependency parser (aka Kurohashi-Nagao Parser), make use of in characterizing the output they produce: both JUMAN and KNP are part of the compression model we build.

Features come in three varieties: semantic, morphological and syntactic. Semantic features are used for classifying entities into semantic types such as name of person, organization, or place, while syntactic features characterize the kinds of dependency

<sup>5</sup>It is worth noting that the present approach can be recast into one based on ‘constraint relaxation’ (Tromble and Eisner, 2006).

<sup>6</sup><http://nlp.kuee.kyoto-u.ac.jp/nl-resource/top-e.html>

relations that hold among BSs such as whether a BS is of the type that combines with the verb (*renyou*), or of the type that combines with the noun (*rentai*), etc.

A morphological feature could be thought of as something that broadly corresponds to an English POS, marking for some syntactic or morphological category such as noun, verb, numeral, etc. Also we included ngram features to encode the lexical context in which a given morpheme appears. Thus we might have something like: for some words (morphemes)  $w_1, w_2$ , and  $w_3$ ,  $f_{w_1 \cdot w_2}(w_3) = 1$  if  $w_3$  is preceded by  $w_1, w_2$ ; otherwise, 0. In addition, we make use of an IR-related feature, whose job is to indicate whether a given morpheme in the input appears in the title of an associated article. The motivation for the feature is obviously to identify concepts relevant to, or unique to the associated article. Also included was a feature on tfidf, to mark words that are conceptually more important than others. The number of features came to around 80,000 for the corpus we used in the experiment.

### 4 The Dependency Path Model

In what follows, we will describe somewhat in detail a prior approach to sentence compression in Japanese which we call the “dependency path model,” or DPM. DPM was first introduced in (Oguro et al., 2000), later explored by a number of people (Morooka et al., 2004; Yamagata et al., 2006; Fukutomi et al., 2007).<sup>7</sup>

DPM has the form:

$$h(\mathbf{y}) = \alpha f(\mathbf{y}) + (1 - \alpha)g(\mathbf{y}), \quad (6)$$

where  $\mathbf{y} = \beta_0, \beta_1, \dots, \beta_{n-1}$ , i.e., a compression consisting of any number of *bunsetsu*’s, or phrase-like elements.  $f(\cdot)$  measures the relevance of content in  $\mathbf{y}$ ; and  $g(\cdot)$  the fluency of text.  $\alpha$  is to provide a way of weighing up contributions from each component.

We further define:

$$f(\mathbf{y}) = \sum_{i=0}^{n-1} q(\beta_i), \quad (7)$$

<sup>7</sup>Kikuchi et al. (2003) explore an approach similar to DPM.

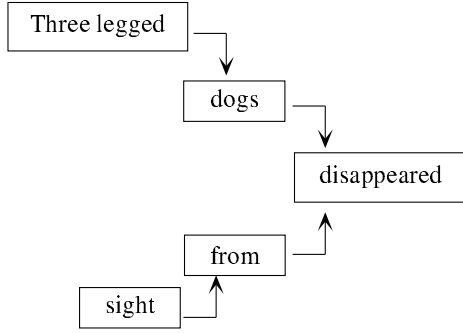


Figure 7: A dependency structure

and

$$g(\mathbf{y}) = \max_s \sum_{i=0}^{n-2} p(\beta_i, \beta_{s(i)}). \quad (8)$$

$q(\cdot)$  is meant to quantify how worthy of inclusion in compression, a given *bunsetsu* is; and  $p(\beta_i, \beta_j)$  represents the connectivity strength of dependency relation between  $\beta_i$  and  $\beta_j$ .  $s(\cdot)$  is a linking function that associates with a *bunsetsu* any one of those that follows it.  $g(\mathbf{y})$  thus represents a set of linked edges that, if combined, give the largest probability for  $\mathbf{y}$ .

*Dependency path length* (DL) refers to the number of (singly linked) dependency relations (or edges) that span two *bunsetsu*'s. Consider the dependency tree in Figure 7, which corresponds to a somewhat contrived sentence 'Three-legged dogs disappeared from sight.' Take an English word for a *bunsetsu* here. We have

$$\begin{aligned} \text{DL}(\text{three-legged}, \text{dogs}) &= 1 \\ \text{DL}(\text{three-legged}, \text{disappeared}) &= 2 \\ \text{DL}(\text{three-legged}, \text{from}) &= \infty \\ \text{DL}(\text{three-legged}, \text{sight}) &= \infty \end{aligned}$$

Since *dogs* is one edge away from *three-legged*, DL for them is 1; and we have DL of two for *three-legged* and *disappeared*, as we need to cross two edges in the direction of arrow to get from the former to the latter. In case there is no path between words as in the last two cases above, we take the DL to be infinite.

DPM takes a dependency tree to be a set of linked edges. Each edge is expressed as a triple  $\langle C_s(\beta_i), C_e(\beta_j), \text{DL}(\beta_i, \beta_j) \rangle$ , where  $\beta_i$  and  $\beta_j$

represent *bunsetsu*'s that the edge spans.  $C_s(\beta)$  denotes the class of a *bunsetsu* where the edge starts and  $C_e(\beta)$  that of a *bunsetsu* where the edge ends. What we mean by 'class of *bunsetsu*' is some sort of a classificatory scheme that concerns linguistic characteristics of *bunsetsu*, such as a part-of-speech of the head, whether it has an inflection, and if it does, what type of inflection it has, etc. Moreover, DPM uses two separate classificatory schemes for  $C_s(\beta)$  and  $C_e(\beta)$ .

In DPM, we define the connectivity strength  $p$  by:

$$p(\beta_i, \beta_j) = \begin{cases} \log S(t) & \text{if } \text{DL}(\beta_i, \beta_j) \neq \infty \\ -\infty & \text{otherwise} \end{cases} \quad (9)$$

where  $t = \langle C_s(\beta_i), C_e(\beta_j), \text{DL}(\beta_i, \beta_j) \rangle$ , and  $S(t)$  is the probability of  $t$  occurring in a compression, which is given by:

$$S(t) = \frac{\# \text{ of } t\text{'s found in compressions}}{\# \text{ of triples found in the training data}} \quad (10)$$

We complete the DPM formulation with:

$$q(\beta) = \log p_c(\beta) + \text{tfidf}(\beta) \quad (11)$$

$p_c(\beta)$  denotes the probability of having *bunsetsu*  $\beta$  in compression, calculated analogously to Eq. 10,<sup>8</sup> and  $\text{tfidf}(\beta)$  obviously denotes the  $\text{tfidf}$  value of  $\beta$ .

In DPM, a compression of a given sentence can be obtained by finding  $\arg \max_{\mathbf{y}} h(\mathbf{y})$ , where  $\mathbf{y}$  ranges over possible candidate compressions of a particular length one may derive from that sentence. In the experiment described later, we set  $\alpha = 0.1$  for DPM, following Morooka et al. (2004), who found the best performance with that setting for  $\alpha$ .

## 5 Evaluation Setup

We created a corpus of sentence summaries based on email news bulletins we had received over five to six months from an on-line news provider called Nikkei Net, which mostly deals with finance and politics.<sup>9</sup> Each bulletin consists of six to seven news briefs, each with a few sentences. Since a news brief contains nothing to indicate what its longer version

<sup>8</sup>DPM puts *bunsetsu*'s into some groups based on linguistic features associated with them, and uses the statistics of the groups for  $p_c$  rather than that of *bunsetsu*'s that actually appear in text.

<sup>9</sup><http://www.nikkei.co.jp>

Table 2: The rating scale on fluency

RATING	EXPLANATION
1	makes no sense
2	only partially intelligible/grammatical
3	makes sense; seriously flawed in grammar
4	makes good sense; only slightly flawed in grammar
5	makes perfect sense; no grammar flaws

might look like, we manually searched the news site for a full-length article that might reasonably be considered a long version of that brief.

We extracted lead sentences both from the brief and from its source article, and aligned them, using what is known as the Smith-Waterman algorithm (Smith and Waterman, 1981), which produced 1,401 pairs of summary and source sentence.<sup>10</sup> For the ease of reference, we call the corpus so produced ‘NICOM’ for the rest of the paper. A part of our system makes use of a modeling toolkit called GRMM (Sutton et al., 2004; Sutton, 2006). Throughout the experiments, we call our approach ‘Generic Sentence Trimmer’ or GST.

## 6 Results and Discussion

We ran DPM and GST on NICOM in the 10-fold cross validation format where we break the data into 10 blocks, use 9 of them for training and test on the remaining block. In addition, we ran the test at three different compression rates, 50%, 60% and 70%, to learn how they affect the way the models perform. This means that for each input sentence in NICOM, we have three versions of its compression created, corresponding to a particular rate at which the sentence is compressed. We call a set of compressions so generated ‘NICOM-g.’

In order to evaluate the quality of outputs GST and DPM generate, we asked 6 people, all Japanese natives, to make an intuitive judgment on how each compression fares in fluency and relevance to gold

<sup>10</sup>The Smith-Waterman algorithm aims at finding a best match between two sequences which may include gaps, such as A-C-D-E and A-B-C-D-E. The algorithm is based on an idea rather akin to dynamic programming.

Table 3: The rating scale on content overlap

RATING	EXPLANATION
1	no overlap with reference
2	poor or marginal overlap w. ref.
3	moderate overlap w. ref.
4	significant overlap w. ref.
5	perfect overlap w. ref.

standards (created by humans), on a scale of 1 to 5. To this end, we conducted evaluation in two separate formats; one concerns fluency and the other relevance. The fluency test consisted of a set of compressions which we created by randomly selecting 200 of them from NICOM-g, for each model at compression rates 50%, 60%, and 70%; thus we have 200 samples for each model and each compression rate.<sup>11</sup> The total number of test compressions came to 1,200.

The relevance test, on the other hand, consisted of *paired* compressions along with the associated gold standard compressions. Each pair contains compressions both from DPM and from GST at a given compression rate. We randomly picked 200 of them from NICOM-g, at each compression rate, and asked the participants to make a subjective judgment on how much of the content in a compression semantically overlap with that of the gold standard, on a scale of 1 to 5 (Table 3). Also included in the survey are 200 gold standard compressions, to get some idea of how fluent “ideal” compressions are, compared to those generated by machine.

Tables 4 and 5 summarize the results. Table 4 looks at the fluency of compressions generated by each of the models; Table 5 looks at how much of the content in reference is retained in compressions. In either table, CR stands for compression rate. All the results are averaged over samples.

We find in Table 4 a clear superiority of GST over DPM at every compression rate examined, with fluency improved by as much as 60% at 60%. However, GST fell short of what human compressions achieved in fluency – an issue we need to address

<sup>11</sup>As stated elsewhere, by compression rate, we mean  $r = \frac{\# \text{ of } 1 \text{ in } y}{\text{length of } x}$ .

Table 4: Fluency (Average)

MODEL/CR	50%	60%	70%
GST	<b>3.430</b>	<b>3.820</b>	<b>3.810</b>
DPM	2.222	2.372	2.660
Human	–	4.45	–

Table 5: Semantic (Content) Overlap (Average)

MODEL/CR	50%	60%	70%
GST	<b>2.720</b>	<b>3.181</b>	<b>3.405</b>
DPM	2.210	2.548	2.890

in the future. Since the average CR of gold standard compressions was 60%, we report their fluency at that rate only.

Table 5 shows the results in relevance of content. Again GST marks a superior performance over DPM, beating it at every compression rate. It is interesting to observe that GST manages to do well in the semantic overlap, despite the cutback on the search space we forced on GST.

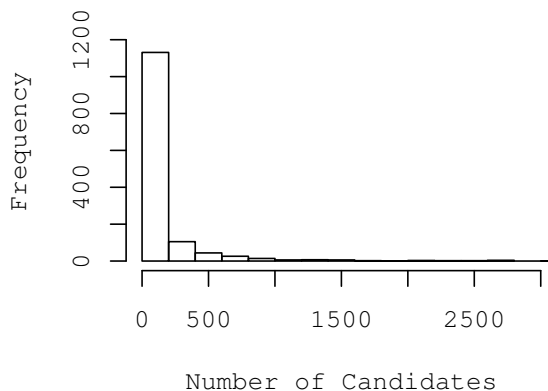
As for fluency, we suspect that the superior performance of GST is largely due to the dependency truncation the model is equipped with; and its performance in content overlap owes a lot to CRFs. However, just how much improvement GST achieved over regular CRFs (with no truncation) in fluency and in relevance is something that remains to be seen, as the latter do not allow for variable length compression, which prohibits a straightforward comparison between the two kinds of models.

We conclude the section with a few words on the size of  $|G(S)|$ , i.e., the number of candidates generated per run of compression with GST.

Figure 8 shows the distribution of the numbers of candidates generated per compression, which looks like the familiar scale-free power curve. Over 99% of the time, the number of candidates or  $|G(S)|$  is found to be less than 500.

## 7 Conclusions

This paper introduced a novel approach to sentence compression in Japanese, which combines a syntactically motivated generation model and CRFs, in or-

Figure 8: The distribution of  $|G(S)|$ 

der to address fluency and relevance of compressions we generate. What distinguishes this work from prior research is its overt withdrawal from a search for global optima to a search for local optima that comply with grammar.

We believe that our idea was empirically borne out, as the experiments found that our approach outperforms, by a large margin, a previously known method called DPM, which employs a global search strategy. The results on semantic overlap indicates that the narrowing down of compressions we search obviously does not harm their relevance to references.

An interesting future exercise would be to explore whether it is feasible to rewrite Eq. 5 as a linear integer program. If it is, the whole scheme of ours would fall under what is known as ‘Linear Programming CRFs’ (Tasker, 2004; Roth and Yih, 2005). What remains to be seen, however, is whether GST is transferable to languages other than Japanese, notably, English. The answer is likely to be yes, but details have yet to be worked out.

## References

James Clarke and Mirella Lapata. 2006. Constraint-based sentence compression: An integer programming

- approach. In *Proceedings of the COLING/ACL 2006*, pages 144–151.
- Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 73–82, Prague, June.
- Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL Text Summarization Workshop and Document Understanding Conference (DUC03)*, pages 1–8, Edmonton, Canada.
- Satoshi Fukutomi, Kazuyuki Takagi, and Kazuhiko Ozeki. 2007. Japanese Sentence Compression using Probabilistic Approach. In *Proceedings of the 13th Annual Meeting of the Association for Natural Language Processing Japan*.
- Michel Galley and Kathleen McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Proceedings of the HLT-NAACL 2007*, pages 180–187.
- Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, pages 310–315.
- Tomonori Kikuchi, Sadaoki Furui, and Chiori Hori. 2003. Two-stage automatic speech summarization by sentence extraction and compaction. In *Proceedings of ICASSP 2003*.
- Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139:91–107.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*.
- Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of the 11th Conference of EACL*, pages 297–304.
- Yuhei Morooka, Makoto Esaki, Kazuyuki Takagi, and Kazuhiko Ozeki. 2004. Automatic summarization of news articles using sentence compaction and extraction. In *Proceedings of the 10th Annual Meeting of Natural Language Processing*, pages 436–439, March. (In Japanese).
- Tadashi Nomoto. 2007. Discriminative sentence compression with conditional random fields. *Information Processing and Management*, 43:1571 – 1587.
- Rei Oguro, Kazuhiko Ozeki, Yujie Zhang, and Kazuyuki Takagi. 2000. An efficient algorithm for Japanese sentence compaction based on phrase importance and inter-phrase dependency. In *Proceedings of TSD 2000 (Lecture Notes in Artificial Intelligence 1902, Springer-Verlag)*, pages 65–81, Brno, Czech Republic.
- Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical functional grammar. In *Proceedings of HLT-NAACL 2003*, pages 118–125, Edmonton.
- Dan Roth and Wen-tau Yih. 2005. Integer linear programming inference for conditional fields. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 05)*.
- T. F. Smith and M. S. Waterman. 1981. Identification of common molecular subsequence. *Journal of Molecular Biology*, 147:195–197.
- Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press. To appear.
- Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic labeling and segmenting sequence data. In *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada.
- Charles Sutton. 2006. GRMM: A graphical models toolkit. <http://mallet.cs.umass.edu>.
- Ben Taskar. 2004. *Learning Structured Prediction Models: A Large Margin Approach*. Ph.D. thesis, Stanford University.
- Roy W. Tromble and Jason Eisner. 2006. A fast finite-state relaxation method for enforcing global constraint on sequence decoding. In *Proceedings of the NAACL*, pages 423–430.
- Jenie Turner and Eugen Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 290–297, Ann Arbor, June.
- Vincent Vandeghinste and Yi Pan. 2004. Sentence compression for automatic subtitling: A hybrid approach. In *Proceedings of the ACL workshop on Text Summarization*, Barcelona.
- Kiwamu Yamagata, Satoshi Fukutomi, Kazuyuki Takagi, and Kazuhiko Ozeki. 2006. Sentence compression using statistical information about dependency path length. In *Proceedings of TSD 2006 (Lecture Notes in Computer Science, Vol. 4188/2006)*, pages 127–134, Brno, Czech Republic.

# A Joint Model of Text and Aspect Ratings for Sentiment Summarization

Ivan Titov

Department of Computer Science  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801  
titov@uiuc.edu

Ryan McDonald

Google Inc.  
76 Ninth Avenue  
New York, NY 10011  
ryanmcd@google.com

## Abstract

Online reviews are often accompanied with numerical ratings provided by users for a set of service or product aspects. We propose a statistical model which is able to discover corresponding topics in text and extract textual evidence from reviews supporting each of these aspect ratings – a fundamental problem in aspect-based sentiment summarization (Hu and Liu, 2004a). Our model achieves high accuracy, without any explicitly labeled data except the user provided opinion ratings. The proposed approach is general and can be used for segmentation in other applications where sequential data is accompanied with correlated signals.

## 1 Introduction

User generated content represents a unique source of information in which user interface tools have facilitated the creation of an abundance of labeled content, e.g., topics in blogs, numerical product and service ratings in user reviews, and helpfulness rankings in online discussion forums. Many previous studies on user generated content have attempted to predict these labels automatically from the associated text. However, these labels are often present in the data already, which opens another interesting line of research: designing models leveraging these labelings to improve a wide variety of applications.

In this study, we look at the problem of *aspect-based sentiment summarization* (Hu and Liu, 2004a; Popescu and Etzioni, 2005; Gamon et al., 2005;

### Nikos' Fine Dining

Food	4/5	"Best fish in the city", "Excellent appetizers"
Decor	3/5	"Cozy with an old world feel", "Too dark"
Service	1/5	"Our waitress was rude", "Awful service"
Value	5/5	"Good Greek food for the \$", "Great price!"

Figure 1: An example aspect-based summary.

Carenini et al., 2006; Zhuang et al., 2006).<sup>1</sup> An aspect-based summarization system takes as input a set of user reviews for a specific product or service and produces a set of relevant aspects, the aggregated sentiment for each aspect, and supporting textual evidence. For example, figure 1 summarizes a restaurant using aspects *food*, *decor*, *service*, and *value* plus a numeric rating out of 5.

Standard aspect-based summarization consists of two problems. The first is *aspect identification and mention extraction*. Here the goal is to find the set of relevant aspects for a rated entity and extract all textual mentions that are associated with each. Aspects can be fine-grained, e.g., *fish*, *lamb*, *calamari*, or coarse-grained, e.g., *food*, *decor*, *service*. Similarly, extracted text can range from a single word to phrases and sentences. The second problem is *sentiment classification*. Once all the relevant aspects and associated pieces of texts are extracted, the system should aggregate sentiment over each aspect to provide the user with an average numeric or symbolic rating. Sentiment classification is a well studied problem (Wiebe, 2000; Pang et al., 2002; Turney, 2002) and in many domains users explicitly

<sup>1</sup>We use the term *aspect* to denote properties of an object that can be rated by a user as in Snyder and Barzilay (2007). Other studies use the term *feature* (Hu and Liu, 2004b).



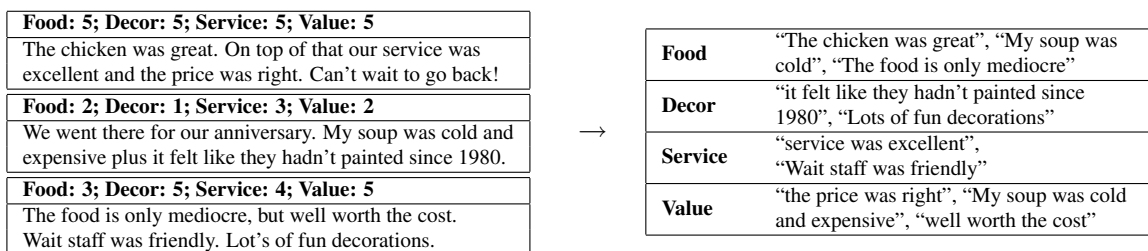


Figure 2: Extraction problem: Produce aspect mentions from a corpus of aspect rated reviews.

provide ratings for each aspect making automated means unnecessary.<sup>2</sup> Aspect identification has also been thoroughly studied (Hu and Liu, 2004b; Gamon et al., 2005; Titov and McDonald, 2008), but again, ontologies and users often provide this information negating the need for automation.

Though it may be reasonable to expect a user to provide a rating for each aspect, it is unlikely that a user will annotate every sentence and phrase in a review as being relevant to some aspect. Thus, it can be argued that the most pressing challenge in an aspect-based summarization system is to extract all relevant mentions for each aspect, as illustrated in figure 2. When labeled data exists, this problem can be solved effectively using a wide variety of methods available for text classification and information extraction (Manning and Schutze, 1999). However, labeled data is often hard to come by, especially when one considers all possible domains of products and services. Instead, we propose an unsupervised model that leverages aspect ratings that frequently accompany an online review.

In order to construct such model, we make two assumptions. First, ratable aspects normally represent coherent topics which can be potentially discovered from co-occurrence information in the text. Second, we hypothesize that the most predictive features of an aspect rating are features derived from the text segments discussing the corresponding aspect. Motivated by these observations, we construct a joint statistical model of text and sentiment ratings. The model is at heart a topic model in that it assigns words to a set of induced *topics*, each of which may represent one particular aspect. The model is extended through a set of maximum entropy classifiers, one per each rated aspect, that are used to pre-

dict the sentiment rating towards each of the aspects. However, only the words assigned to an aspects corresponding topic are used in predicting the rating for that aspect. As a result, the model enforces that words assigned to an aspects' topic are predictive of the associated rating. Our approach is more general than the particular statistical model we consider in this paper. For example, other topic models can be used as a part of our model and the proposed class of models can be employed in other tasks beyond sentiment summarization, e.g., segmentation of blogs on the basis of topic labels provided by users, or topic discovery on the basis of tags given by users on social bookmarking sites.<sup>3</sup>

The rest of the paper is structured as follows. Section 2 begins with a discussion of the joint text-sentiment model approach. In Section 3 we provide both a qualitative and quantitative evaluation of the proposed method. We conclude in Section 4 with an examination of related work.

## 2 The Model

In this section we describe a new statistical model called the Multi-Aspect Sentiment model (MAS), which consists of two parts. The first part is based on Multi-Grain Latent Dirichlet Allocation (Titov and McDonald, 2008), which has been previously shown to build topics that are representative of ratable aspects. The second part is a set of sentiment predictors per aspect that are designed to force specific topics in the model to be directly correlated with a particular aspect.

### 2.1 Multi-Grain LDA

The Multi-Grain Latent Dirichlet Allocation model (MG-LDA) is an extension of Latent Dirichlet Allocation (LDA) (Blei et al., 2003). As was demon-

<sup>2</sup>E.g., <http://zagat.com> and <http://tripadvisor.com>.

<sup>3</sup>See e.g. [del.icio.us](http://del.icio.us) (<http://del.icio.us>).

strated in Titov and McDonald (2008), the topics produced by LDA do not correspond to ratable aspects of entities. In particular, these models tend to build topics that globally classify terms into product instances (e.g., Creative Labs Mp3 players versus iPods, or New York versus Paris Hotels). To combat this, MG-LDA models two distinct types of topics: global topics and local topics. As in LDA, the distribution of global topics is fixed for a document (a user review). However, the distribution of local topics is allowed to vary across the document.

A word in the document is sampled either from the mixture of global topics or from the mixture of local topics specific to the local context of the word. It was demonstrated in Titov and McDonald (2008) that ratable aspects will be captured by local topics and global topics will capture properties of reviewed items. For example, consider an extract from a review of a London hotel: “. . . public transport in London is straightforward, the tube station is about an 8 minute walk . . . or you can get a bus for £1.50”. It can be viewed as a mixture of topic *London* shared by the entire review (words: “London”, “tube”, “£”), and the ratable aspect *location*, specific for the local context of the sentence (words: “transport”, “walk”, “bus”). Local topics are reused between very different types of items, whereas global topics correspond only to particular types of items.

In MG-LDA a document is represented as a set of sliding windows, each covering  $T$  adjacent sentences within a document.<sup>4</sup> Each window  $v$  in document  $d$  has an associated distribution over local topics  $\theta_{d,v}^{loc}$  and a distribution defining preference for local topics versus global topics  $\pi_{d,v}$ . A word can be sampled using any window covering its sentence  $s$ , where the window is chosen according to a categorical distribution  $\psi_{d,s}$ . Importantly, the fact that windows overlap permits the model to exploit a larger co-occurrence domain. These simple techniques are capable of modeling local topics without more expensive modeling of topic transitions used in (Griffiths et al., 2004; Wang and McCallum, 2005; Wallach, 2006; Gruber et al., 2007). Introduction of a symmetrical Dirichlet prior  $Dir(\gamma)$  for the distribution  $\psi_{d,s}$  can control the smoothness of transitions.

<sup>4</sup>Our particular implementation is over sentences, but sliding windows in theory can be over any sized fragment of text.

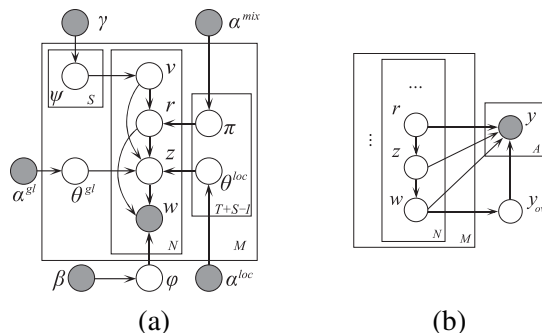


Figure 3: (a) MG-LDA model. (b) An extension of MG-LDA to obtain MAS.

The formal definition of the model with  $K^{gl}$  global and  $K^{loc}$  local topics is as follows: First, draw  $K^{gl}$  word distributions for global topics  $\varphi_z^{gl}$  from a Dirichlet prior  $Dir(\beta^{gl})$  and  $K^{loc}$  word distributions for local topics  $\varphi_z^{loc}$  - from  $Dir(\beta^{loc})$ . Then, for each document  $d$ :

- Choose a distribution of global topics  $\theta_d^{gl} \sim Dir(\alpha^{gl})$ .
- For each sentence  $s$  choose a distribution over sliding windows  $\psi_{d,s}(v) \sim Dir(\gamma)$ .
- For each sliding window  $v$ 
  - choose  $\theta_{d,v}^{loc} \sim Dir(\alpha^{loc})$ ,
  - choose  $\pi_{d,v} \sim Beta(\alpha^{mix})$ .
- For each word  $i$  in sentence  $s$  of document  $d$ 
  - choose window  $v_{d,i} \sim \psi_{d,s}$ ,
  - choose  $r_{d,i} \sim \pi_{d,v_{d,i}}$ ,
  - if  $r_{d,i} = gl$  choose global topic  $z_{d,i} \sim \theta_d^{gl}$ ,
  - if  $r_{d,i} = loc$  choose local topic  $z_{d,i} \sim \theta_{d,v_{d,i}}^{loc}$ ,
  - choose word  $w_{d,i}$  from the word distribution  $\varphi_{z_{d,i}}^{r_{d,i}}$ .

$Beta(\alpha^{mix})$  is a prior Beta distribution for choosing between local and global topics. In Figure 3a the corresponding graphical model is presented.

## 2.2 Multi-Aspect Sentiment Model

MG-LDA constructs a set of topics that ideally correspond to ratable aspects of an entity (often in a many-to-one relationship of topics to aspects). A major shortcoming of this model – and all other unsupervised models – is that this correspondence is not explicit, i.e., how does one say that topic  $X$  is really about aspect  $Y$ ? However, we can observe that numeric aspect ratings are often included in our data by users who left the reviews. We then make the assumption that the text of the review discussing an aspect is predictive of its rating. Thus, if we model the prediction of aspect ratings jointly with the construction of explicitly associated topics, then such a

model should benefit from both higher quality topics and a direct assignment from topics to aspects. This is the basic idea behind the Multi-Aspect Sentiment model (MAS).

In its simplest form, MAS introduces a classifier for each aspect, which is used to predict its rating. Each classifier is explicitly associated to a single topic in the model and only words assigned to that topic can participate in the prediction of the sentiment rating for the aspect. However, it has been observed that ratings for different aspects can be correlated (Snyder and Barzilay, 2007), e.g., very negative opinion about room cleanliness is likely to result not only in a low rating for the aspect *rooms*, but also is very predictive of low ratings for the aspects *service* and *dining*. This complicates discovery of the corresponding topics, as in many reviews the most predictive features for an aspect rating might correspond to another aspect. Another problem with this overly simplistic model is the presence of opinions about an item in general without referring to any particular aspect. For example, “this product is the worst I have ever purchased” is a good predictor of low ratings for every aspect. In such cases, non-aspect ‘background’ words will appear to be the most predictive. Therefore, the use of the aspect sentiment classifiers based only on the words assigned to the corresponding topics is problematic. Such a model will not be able to discover coherent topics associated with each aspect, because in many cases the most predictive fragments for each aspect rating will not be the ones where this aspect is discussed.

Our proposal is to estimate the distribution of possible values of an aspect rating on the basis of the *overall* sentiment rating and to use the words assigned to the corresponding topic to compute *corrections* for this aspect. An aspect rating is typically correlated to the overall sentiment rating<sup>5</sup> and the fragments discussing this particular aspect will help to correct the overall sentiment in the appropriate direction. For example, if a review of a hotel is generally positive, but it includes a sentence “the neighborhood is somewhat seedy” then this sentence is predictive of rating for an aspect *location* being below other ratings. This rectifies the aforementioned

<sup>5</sup>In the dataset used in our experiments all three aspect ratings are equivalent for 5,250 reviews out of 10,000.

problems. First, aspect sentiment ratings can often be regarded as conditionally independent given the overall rating, therefore the model will not be forced to include in an aspect topic any words from other aspect topics. Secondly, the fragments discussing overall opinion will influence the aspect rating only through the overall sentiment rating. The overall sentiment is almost always present in the real data along with the aspect ratings, but it can be coarsely discretized and we preferred to use a latent overall sentiment.

The MAS model is presented in Figure 3b. Note that for simplicity we decided to omit in the figure the components of the MG-LDA model other than variables  $r$ ,  $z$  and  $w$ , though they are present in the statistical model. MAS also allows for extra unassociated local topics in order to capture aspects not explicitly rated by the user. As in MG-LDA, MAS has global topics which are expected to capture topics corresponding to particular types of items, such *London hotels* or *seaside resorts* for the hotel domain. In figure 3b we shaded the aspect ratings  $y_a$ , assuming that every aspect rating is present in the data (though in practice they might be available only for some reviews). In this model the distribution of the overall sentiment rating  $y_{ov}$  is based on all the n-gram features of a review text. Then the distribution of  $y_a$ , for every rated aspect  $a$ , can be computed from the distribution of  $y_{ov}$  and from any n-gram feature where at least one word in the n-gram is assigned to the associated aspect topic ( $r = loc$ ,  $z = a$ ).

Instead of having a latent variable  $y_{ov}$ ,<sup>6</sup> we use a similar model which does not have an explicit notion of  $y_{ov}$ . The distribution of a sentiment rating  $y_a$  for each rated aspect  $a$  is computed from two scores. The first score is computed on the basis of all the n-grams, but using a common set of weights independent of the aspect  $a$ . Another score is computed only using n-grams associated with the related topic, but an aspect-specific set of weights is used in this computation. More formally, we consider the log-linear distribution:

$$P(y_a = y | \mathbf{w}, \mathbf{r}, \mathbf{z}) \propto \exp(b_y^a + \sum_{f \in \mathbf{w}} J_{f,y}^a + p_{f,\mathbf{r},\mathbf{z}}^a J_{f,y}^a), \quad (1)$$

where  $\mathbf{w}$ ,  $\mathbf{r}$ ,  $\mathbf{z}$  are vectors of all the words in a docu-

<sup>6</sup>Preliminary experiments suggested that this is also a feasible approach, but somewhat more computationally expensive.

ment, assignments of context (global or local) and topics for all the words in the document, respectively.  $b_y^a$  is the bias term which regulates the prior distribution  $P(y_a = y)$ ,  $f$  iterates through all the n-grams,  $J_{y,f}$  and  $J_{y,f}^a$  are common weights and aspect-specific weights for n-gram feature  $f$ .  $p_{f,r,z}^a$  is equal to a fraction of words in n-gram feature  $f$  assigned to the aspect topic ( $r = loc, z = a$ ).

### 2.3 Inference in MAS

Exact inference in the MAS model is intractable. Following Titov and McDonald (2008) we use a collapsed Gibbs sampling algorithm that was derived for the MG-LDA model based on the Gibbs sampling method proposed for LDA in (Griffiths and Steyvers, 2004). Gibbs sampling is an example of a Markov Chain Monte Carlo algorithm (Geman and Geman, 1984). It is used to produce a sample from a joint distribution when only conditional distributions of each variable can be efficiently computed. In Gibbs sampling, variables are sequentially sampled from their distributions conditioned on all other variables in the model. Such a chain of model states converges to a sample from the joint distribution. A naive application of this technique to LDA would imply that both assignments of topics to words  $\mathbf{z}$  and distributions  $\theta$  and  $\varphi$  should be sampled. However, (Griffiths and Steyvers, 2004) demonstrated that an efficient collapsed Gibbs sampler can be constructed, where only assignments  $\mathbf{z}$  need to be sampled, whereas the dependency on distributions  $\theta$  and  $\varphi$  can be integrated out analytically.

In the case of MAS we also use maximum a-posteriori estimates of the sentiment predictor parameters  $b_y^a$ ,  $J_{y,f}$  and  $J_{y,f}^a$ . The MAP estimates for parameters  $b_y^a$ ,  $J_{y,f}$  and  $J_{y,f}^a$  are obtained by using stochastic gradient ascent. The direction of the gradient is computed simultaneously with running a chain by generating several assignments at each step and averaging over the corresponding gradient estimates. For details on computing gradients for log-linear graphical models with Gibbs sampling we refer the reader to (Neal, 1992).

Space constraints do not allow us to present either the derivation or a detailed description of the sampling algorithm. However, note that the conditional distribution used in sampling decomposes into two

parts:

$$P(v_{d,i} = v, r_{d,i} = r, z_{d,i} = z | \mathbf{v}', \mathbf{r}', \mathbf{z}', \mathbf{w}, \mathbf{y}) \propto \eta_{v,r,z}^{d,i} \times \rho_{r,z}^{d,i}, \quad (2)$$

where  $\mathbf{v}'$ ,  $\mathbf{r}'$  and  $\mathbf{z}'$  are vectors of assignments of sliding windows, context (global or local) and topics for all the words in the collection except for the considered word at position  $i$  in document  $d$ ;  $y$  is the vector of sentiment ratings. The first factor  $\eta_{v,r,z}^{d,i}$  is responsible for modeling co-occurrences on the window and document level and coherence of the topics. This factor is proportional to the conditional distribution used in the Gibbs sampler of the MG-LDA model (Titov and McDonald, 2008). The last factor quantifies the influence of the assignment of the word  $(d, i)$  on the probability of the sentiment ratings. It appears only if ratings are known (observable) and equals:

$$\rho_{r,z}^{d,i} = \prod_a \frac{P(y_a^d | \mathbf{w}, \mathbf{r}', r_{d,i} = r, \mathbf{z}', z_{d,i} = z)}{P(y_a^d | \mathbf{w}, \mathbf{r}', \mathbf{z}', r_{d,i} = gl)},$$

where the probability distribution is computed as defined in expression (1),  $y_a^d$  is the rating for the  $a$ th aspect of review  $d$ .

## 3 Experiments

In this section we present qualitative and quantitative experiments. For the qualitative analysis we show that topics inferred by the MAS model correspond directly to the associated aspects. For the quantitative analysis we show that the MAS model induces a distribution over the rated aspects which can be used to accurately predict whether a text fragment is relevant to an aspect or not.

### 3.1 Qualitative Evaluation

To perform qualitative experiments we used a set of reviews of hotels taken from TripAdvisor.com<sup>7</sup> that contained 10,000 reviews (109,024 sentences, 2,145,313 words in total). Every review was rated with at least three aspects: *service*, *location* and *rooms*. Each rating is an integer from 1 to 5. The dataset was tokenized and sentence split automatically.

<sup>7</sup>(c) 2005-06, TripAdvisor, LLC All rights reserved

	rated aspect	top words
local topics	service	staff friendly helpful service desk concierge excellent extremely hotel great reception english pleasant help
	location	hotel walk location station metro walking away right minutes close bus city located just easy restaurants
	rooms	room bathroom shower bed tv small water clean comfortable towels bath nice large pillows space beds tub
	-	breakfast free coffee internet morning access buffet day wine nice lobby complimentary included good fruit
global topics	-	\$ night parking rate price paid day euros got cost pay hotel worth euro expensive car extra deal booked
	-	room noise night street air did door floor rooms open noisy window windows hear outside problem quiet sleep
	-	moscow st russian petersburg nevsky russia palace hermitage kremlin prospect river prospekt kempinski
		paris tower french eiffel dame notre rue st louvre rer champs opera elysee george parisian du pantheon cafes

Table 1: Top words from MAS for hotel reviews.

$K_{rooms}$	top words
2	rooms clean hotel room small nice comfortable modern good quite large lobby old decor spacious decorated bathroom size room noise night street did air rooms door open noisy window floor hear windows problem outside quiet sleep bit light
3	room clean bed comfortable rooms bathroom small beds nice large size tv spacious good double big space huge king room floor view rooms suite got views given quiet building small balcony upgraded nice high booked asked overlooking room bathroom shower air water did like hot small towels door old window toilet conditioning open bath dirty wall tub
4	room clean rooms comfortable bed small beds nice bathroom size large modern spacious good double big quiet decorated check arrived time day airport early room luggage took late morning got long flight ready minutes did taxi bags went room noise night street did air rooms noisy open door hear windows window outside quiet sleep problem floor conditioning bathroom room shower tv bed small water towels bath tub large nice toilet clean space toiletries flat wall sink screen

Table 2: Top words for aspect *rooms* with different number of topics  $K_{rooms}$ .

We ran the sampling chain for 700 iterations to produce a sample. Distributions of words in each topic were estimated as the proportion of words assigned to each topic, taking into account topic model priors  $\beta^{gl}$  and  $\beta^{loc}$ . The sliding windows were chosen to cover 3 sentences for all the experiments. All the priors were chosen to be equal to 0.1. We used 15 local topics and 30 global topics. In the model, the first three local topics were associated to the rating classifiers for each aspects. As a result, we would expect these topics to correspond to the *service*, *location*, and *rooms* aspects respectively. Unigram and bigram features were used in the sentiment predictors in the MAS model. Before applying the topic models we removed punctuation and also removed stop words using the standard list of stop words,<sup>8</sup> however, all the words and punctuation were used in the sentiment predictors.

It does not take many chain iterations to discover initial topics. This happens considerably faster than the appropriate weights of the sentiment predictor being learned. This poses a problem, because, in the beginning, the sentiment predictors are not accurate enough to force the model to discover appropriate topics associated with each of the rated aspects. And as soon as topic are formed, aspect sentiment predictors cannot affect them anymore because they do not

<sup>8</sup>[http://www.dcs.gla.ac.uk/idom/ir\\_resources/linguistic\\_utils/stop\\_words](http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words)

have access to the true words associated with their aspects. To combat this problem we first train the sentiment classifiers by assuming that  $p_{f,r,z}^a$  is equal for all the local topics, which effectively ignores the topic model. Then we use the estimated parameters within the topic model.<sup>9</sup> Secondly, we modify the sampling algorithm. The conditional probability used in sampling, expression (2), is proportional to the product of two factors. The first factor,  $\eta_{v,r,z}^{d,i}$ , expresses a preference for topics likely from the co-occurrence information, whereas the second one,  $\rho_{r,z}^{d,i}$ , favors the choice of topics which are predictive of the observable sentiment ratings. We used  $(\rho_{r,z}^{d,i})^{1+0.95^t q}$  in the sampling distribution instead of  $\rho_{r,z}^{d,i}$ , where  $t$  is the iteration number.  $q$  was chosen to be 4, though the quality of the topics seemed to be indistinguishable with any  $q$  between 3 and 10. This can be thought of as having  $1 + 0.95^t q$  ratings instead of a single vector assigned to each review, i.e., focusing the model on prediction of the ratings rather than finding the topic labels which are good at explaining co-occurrences of words. These heuristics influence sampling only during the first iterations of the chain.

Top words for some of discovered local topics, in-

<sup>9</sup>Initial experiments suggested that instead of doing this ‘pre-training’ we could start with very large priors  $\alpha^{loc}$  and  $\alpha^{mix}$ , and then reduce them through the course of training. However, this is significantly more computationally expensive.

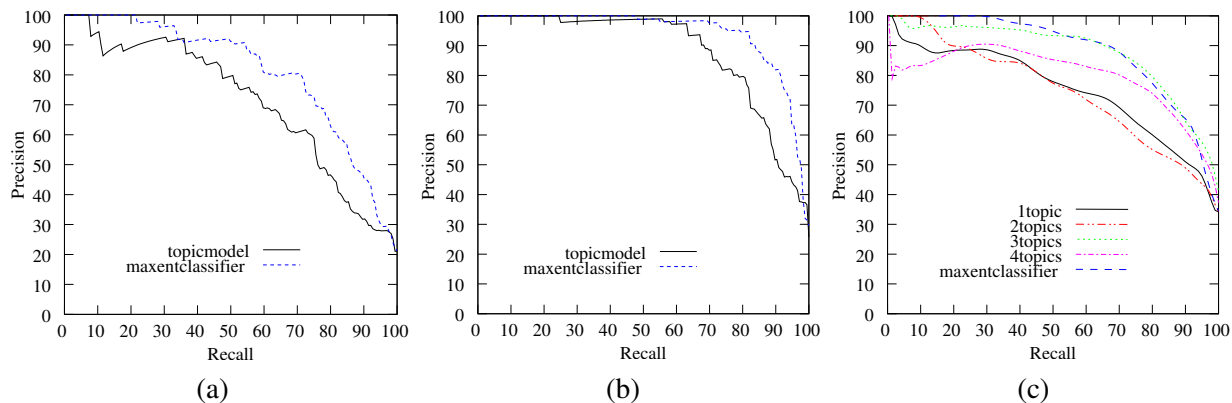


Figure 4: (a) Aspect *service*. (b) Aspect *location*. (c) Aspect *rooms*.

cluding the first 3 topics associated with the rated aspects, and also top words for some of global topics are presented in Table 1. We can see that the model discovered as its first three topics the correct associated aspects: *service*, *location*, and *rooms*. Other local topics, as for the MG-LDA model, correspond to other aspects discussed in reviews (*breakfast*, *prices*, *noise*), and as it was previously shown in Titov and McDonald (2008), aspects for global topics correspond to the types of reviewed items (*hotels in Russia*, *Paris hotels*) or background words.

Notice though, that the 3rd local topic induced for the rating *rooms* is slightly narrow. This can be explained by the fact that the aspect *rooms* is a central aspect of hotel reviews. A very significant fraction of text in every review can be thought of as a part of the aspect *rooms*. These portions of reviews discuss different coherent sub-aspects related to the aspect *rooms*, e.g., the previously discovered topic *noise*. Therefore, it is natural to associate several topics to such central aspects. To test this we varied the number of topics associated with the sentiment predictor for the aspect *rooms*. Top words for resulting topics are presented in Table 2. It can be observed that the topic model discovered appropriate topics while the number of topics was below 4. With 4 topics a semantically unrelated topic (*check-in/arrival*) is induced. Manual selection of the number of topics is undesirable, but this problem can be potentially tackled with Dirichlet Process priors or a topic split criterion based on the accuracy of the sentiment predictor in the MAS model. We found that both *service* and *location* did not benefit by the assignment of additional topics to their sentiment rating models.

The experimental results suggest that the MAS model is reliable in the discovery of topics corresponding to the rated aspects. In the next section we will show that the induced topics can be used to accurately extract fragments for each aspect.

### 3.2 Sentence Labeling

A primary advantage of MAS over unsupervised models, such as MG-LDA or clustering, is that topics are linked to a rated aspect, i.e., we know exactly which topics model which aspects. As a result, these topics can be directly used to extract textual mentions that are relevant for an aspect. To test this, we hand labeled 779 random sentences from the dataset considered in the previous set of experiments. The sentences were labeled with one or more aspects. Among them, 164, 176 and 263 sentences were labeled as related to aspects *service*, *location* and *rooms*, respectively. The remaining sentences were not relevant to any of the rated aspects.

We compared two models. The first model uses the first three topics of MAS to extract relevant mentions based on the probability of that topic/aspect being present in the sentence. To obtain these probabilities we used estimators based on the proportion of words in the sentence assigned to an aspects' topic and normalized within local topics. To improve the reliability of the estimator we produced 100 samples for each document while keeping assignments of the topics to all other words in the collection fixed. The probability estimates were then obtained by averaging over these samples. We did not perform any model selection on the basis of the hand-labeled data, and tested only a single model of each type.

For the second model we trained a maximum entropy classifier, one per each aspect, using 10-fold cross validation and unigram/bigram features. Note that this is a *supervised* system and as such represents an upper-bound in performance one might expect when comparing an unsupervised model such as MAS. We chose this comparison to demonstrate that our model can find relevant text mentions with high accuracy relative to a supervised model. It is difficult to compare our model to other unsupervised systems such as MG-LDA or LDA. Again, this is because those systems have no mechanism for directly correlating topics or clusters to corresponding aspects, highlighting the benefit of MAS.

The resulting precision-recall curves for the aspects *service*, *location* and *rooms* are presented in Figure 4. In Figure 4c, we varied the number of topics associated with the aspect *rooms*.<sup>10</sup> The average precision we obtained (the standard measure proportional to the area under the curve) is 75.8%, 85.5% for aspects *service* and *location*, respectively. For the aspect *rooms* these scores are equal to 75.0%, 74.5%, 87.6%, 79.8% with 1–4 topics per aspect, respectively. The logistic regression models achieve 80.8%, 94.0% and 88.3% for the aspects *service*, *location* and *rooms*. We can observe that the topic model, which does not use any explicitly aspect-labeled text, achieves accuracies lower than, but comparable to a supervised model.

## 4 Related Work

There is a growing body of work on summarizing sentiment by extracting and aggregating sentiment over ratable aspects and providing corresponding textual evidence. Text excerpts are usually extracted through string matching (Hu and Liu, 2004a; Popescu and Etzioni, 2005), sentence clustering (Gamon et al., 2005), or through topic models (Mei et al., 2007; Titov and McDonald, 2008). String extraction methods are limited to fine-grained aspects whereas clustering and topic model approaches must resort to ad-hoc means of labeling clusters or topics. However, this is the first work we are aware of that uses a pre-defined set of aspects plus an associated signal to learn a mapping from text to an aspect for

<sup>10</sup>To improve readability we smoothed the curve for the aspect *rooms*.

the purpose of extraction.

A closely related model to ours is that of Mei et al. (2007) which performs joint topic and sentiment modeling of collections. Our model differs from theirs in many respects: Mei et al. only model sentiment predictions for the entire document and not on the aspect level; They treat sentiment predictions as unobserved variables, whereas we treat them as observed signals that help to guide the creation of topics; They model co-occurrences solely on the document level, whereas our model is based on MG-LDA and models both local and global contexts.

Recently, Blei and McAuliffe (2008) proposed an approach for joint sentiment and topic modeling that can be viewed as a supervised LDA (sLDA) model that tries to infer topics appropriate for use in a given classification or regression problem. MAS and sLDA are similar in that both use sentiment predictions as an observed signal that is predicted by the model. However, Blei et al. do not consider multi-aspect ranking or look at co-occurrences beyond the document level, both of which are central to our model. Parallel to this study Branavan et al. (2008) also showed that joint models of text and user annotations benefit extractive summarization. In particular, they used signals from pros-cons lists whereas our models use aspect rating signals.

## 5 Conclusions

In this paper we presented a joint model of text and aspect ratings for extracting text to be displayed in sentiment summaries. The model uses aspect ratings to discover the corresponding topics and can thus extract fragments of text discussing these aspects without the need of annotated data. We demonstrated that the model indeed discovers corresponding coherent topics and achieves accuracy in sentence labeling comparable to a standard supervised model. The primary area of future work is to incorporate the model into an end-to-end sentiment summarization system in order to evaluate it at that level.

## Acknowledgments

This work benefited from discussions with Sasha Blair-Goldensohn and Fernando Pereira.

## References

- David M. Blei and Jon D. McAuliffe. 2008. Supervised topic models. In *Advances in Neural Information Processing Systems (NIPS)*.
- D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(5):993–1022.
- S.R.K. Branavan, H. Chen, J. Eisenstein, and R. Barzilay. 2008. Learning document-level semantic properties from free-text annotations. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.
- G. Carenini, R. Ng, and A. Pauls. 2006. Multi-Document Summarization of Evaluative Text. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*.
- M. Gamon, A. Aue, S. Corston-Oliver, and E. Ringger. 2005. Pulse: Mining customer opinions from free text. In *Proc. of the 6th International Symposium on Intelligent Data Analysis*, pages 121–132.
- S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.
- T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101 Suppl 1:5228–5235.
- T. L. Griffiths, M. Steyvers, D. M. Blei, and J. B. Tenenbaum. 2004. Integrating topics and syntax. In *Advances in Neural Information Processing Systems*.
- A. Gruber, Y. Weiss, and M. Rosen-Zvi. 2007. Hidden Topic Markov Models. In *Proceedings of the Conference on Artificial Intelligence and Statistics*.
- M. Hu and B. Liu. 2004a. Mining and summarizing customer reviews. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM Press New York, NY, USA.
- M. Hu and B. Liu. 2004b. Mining Opinion Features in Customer Reviews. In *Proceedings of Nineteenth National Conference on Artificial Intelligence*.
- C. Manning and M. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.
- Q. Mei, X. Ling, M. Wondra, H. Su, and C.X. Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th International Conference on World Wide Web*, pages 171–180.
- Radford Neal. 1992. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113.
- B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- A.M. Popescu and O. Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- B. Snyder and R. Barzilay. 2007. Multiple Aspect Ranking using the Good Grief Algorithm. In *Proceedings of the Joint Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies*, pages 300–307.
- I. Titov and R. McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International Conference on World Wide Web*.
- P. Turney. 2002. Thumbs up or thumbs down? Sentiment orientation applied to unsupervised classification of reviews. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.
- Hanna M. Wallach. 2006. Topic modeling; beyond bag of words. In *International Conference on Machine Learning*.
- Xuerui Wang and Andrew McCallum. 2005. A note on topical n-grams. Technical Report UM-CS-2005-071, University of Massachusetts.
- J. Wiebe. 2000. Learning subjective adjectives from corpora. In *Proceedings of the National Conference on Artificial Intelligence*.
- L. Zhuang, F. Jing, and X.Y. Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM)*, pages 43–50.



# Improving Parsing and PP attachment Performance with Sense Information

**Eneko Agirre**  
IXA NLP Group  
University of the Basque Country  
Donostia, Basque Country  
e.agirre@ehu.es

**Timothy Baldwin**  
LT Group, CSSE  
University of Melbourne  
Victoria 3010 Australia  
tim@csse.unimelb.edu.au

**David Martinez**  
LT Group, CSSE  
University of Melbourne  
Victoria 3010 Australia  
davidm@csse.unimelb.edu.au

## Abstract

To date, parsers have made limited use of semantic information, but there is evidence to suggest that semantic features can enhance parse disambiguation. This paper shows that semantic classes help to obtain significant improvement in both parsing and PP attachment tasks. We devise a gold-standard sense- and parse tree-annotated dataset based on the intersection of the Penn Treebank and SemCor, and experiment with different approaches to both semantic representation and disambiguation. For the Bikel parser, we achieved a maximal error reduction rate over the baseline parser of 6.9% and 20.5%, for parsing and PP-attachment respectively, using an unsupervised WSD strategy. This demonstrates that word sense information can indeed enhance the performance of syntactic disambiguation.

## 1 Introduction

Traditionally, parse disambiguation has relied on structural features extracted from syntactic parse trees, and made only limited use of semantic information. There is both empirical evidence and linguistic intuition to indicate that semantic features can enhance parse disambiguation performance, however. For example, a number of different parsers have been shown to benefit from lexicalisation, that is, the conditioning of structural features on the lexical head of the given constituent (Magerman, 1995; Collins, 1996; Charniak, 1997; Charniak, 2000; Collins, 2003). As an example of lexicalisation, we may observe in our training data that *knife* often occurs as the manner adjunct of *open* in prepositional phrases headed by *with* (c.f. *open with*

*a knife*), which would provide strong evidence for *with (a) knife* attaching to *open* and not *box* in *open the box with a knife*. It would not, however, provide any insight into the correct attachment of *with scissors* in *open the box with scissors*, as the disambiguation model would not be able to predict that *knife* and *scissors* are semantically similar and thus likely to have the same attachment preferences.

In order to deal with this limitation, we propose to integrate directly the semantic classes of words into the process of training the parser. This is done by substituting the original words with semantic codes that reflect semantic classes. For example, in the above example we could substitute both *knife* and *scissors* with the semantic class TOOL, thus relating the training and test instances directly. We explore several models for semantic representation, based around WordNet (Fellbaum, 1998).

Our approach to exploring the impact of lexical semantics on parsing performance is to take two state-of-the-art statistical treebank parsers and preprocess the inputs variously. This simple method allows us to incorporate semantic information into the parser without having to reimplement a full statistical parser, and also allows for maximum comparability with existing results in the treebank parsing community. We test the parsers over both a PP attachment and full parsing task.

In experimenting with different semantic representations, we require some strategy to disambiguate the semantic class of polysemous words in context (e.g. determining for each instance of *crane* whether it refers to an animal or a lifting device). We explore a number of disambiguation strategies, including the use of hand-annotated (gold-standard) senses, the

use of the most frequent sense, and an unsupervised word sense disambiguation (WSD) system.

This paper shows that semantic classes help to obtain significant improvements for both PP attachment and parsing. We attain a 20.5% error reduction for PP attachment, and 6.9% for parsing. These results are achieved using most frequent sense information, which surprisingly outperforms both gold-standard senses and automatic WSD.

The results are notable in demonstrating that very simple preprocessing of the parser input facilitates significant improvements in parser performance. We provide the first definitive results that word sense information can enhance Penn Treebank parser performance, building on earlier results of Bikel (2000) and Xiong et al. (2005). Given our simple procedure for incorporating lexical semantics into the parsing process, our hope is that this research will open the door to further gains using more sophisticated parsing models and richer semantic options.

## 2 Background

This research is focused on applying lexical semantics in parsing and PP attachment tasks. Below, we outline these tasks.

### Parsing

As our baseline parsers, we use two state-of-the-art lexicalised parsing models, namely the Bikel parser (Bikel, 2004) and Charniak parser (Charniak, 2000). While a detailed description of the respective parsing models is beyond the scope of this paper, it is worth noting that both parsers induce a context free grammar as well as a generative parsing model from a training set of parse trees, and use a development set to tune internal parameters. Traditionally, the two parsers have been trained and evaluated over the WSJ portion of the Penn Treebank (PTB; Marcus et al. (1993)). We diverge from this norm in focusing exclusively on a sense-annotated subset of the Brown Corpus portion of the Penn Treebank, in order to investigate the upper bound performance of the models given gold-standard sense information.

### PP attachment in a parsing context

Prepositional phrase attachment (PP attachment) is the problem of determining the correct attachment site for a PP, conventionally in the form of the noun

or verb in a V NP PP structure (Ratnaparkhi et al., 1994; Mitchell, 2004). For instance, in *I ate a pizza with anchovies*, the PP *with anchovies* could attach either to the verb (c.f. *ate with anchovies*) or to the noun (c.f. *pizza with anchovies*), of which the noun is the correct attachment site. With *I ate a pizza with friends*, on the other hand, the verb is the correct attachment site. PP attachment is a structural ambiguity problem, and as such, a subproblem of parsing.

Traditionally the so-called RRR data (Ratnaparkhi et al., 1994) has been used to evaluate PP attachment algorithms. RRR consists of 20,081 training and 3,097 test quadruples of the form  $(v, n1, p, n2)$ , where the attachment decision is either  $v$  or  $n1$ . The best published results over RRR are those of Stetina and Nagao (1997), who employ WordNet sense predictions from an unsupervised WSD method within a decision tree classifier. Their work is particularly inspiring in that it significantly outperformed the plethora of lexicalised probabilistic models that had been proposed to that point, and has not been beaten in later attempts.

In a recent paper, Atterer and Schütze (2007) criticised the RRR dataset because it assumes that an oracle parser provides the two hypothesised structures to choose between. This is needed to derive the fact that there are two possible attachment sites, as well as information about the lexical phrases, which are typically extracted heuristically from gold standard parses. Atterer and Schütze argue that the only meaningful setting for PP attachment is within a parser, and go on to demonstrate that in a parser setting, the Bikel parser is competitive with the best-performing dedicated PP attachment methods. Any improvement in PP attachment performance over the baseline Bikel parser thus represents an advancement in state-of-the-art performance.

That we specifically present results for PP attachment in a parsing context is a combination of us supporting the new research direction for PP attachment established by Atterer and Schütze, and us wishing to reinforce the findings of Stetina and Nagao that word sense information significantly enhances PP attachment performance in this new setting.

### Lexical semantics in parsing

There have been a number of attempts to incorporate word sense information into parsing tasks. The

most closely related research is that of Bikel (2000), who merged the Brown portion of the Penn Treebank with SemCor (similarly to our approach in Section 4.1), and used this as the basis for evaluation of a generative bilexical model for joint WSD and parsing. He evaluated his proposed model in a parsing context both with and without WordNet-based sense information, and found that the introduction of sense information either had no impact or degraded parse performance.

The only successful applications of word sense information to parsing that we are aware of are Xiong et al. (2005) and Fujita et al. (2007). Xiong et al. (2005) experimented with first-sense and hypernym features from HowNet and CiLin (both WordNets for Chinese) in a generative parse model applied to the Chinese Penn Treebank. The combination of word sense and first-level hypernyms produced a significant improvement over their basic model. Fujita et al. (2007) extended this work in implementing a discriminative parse selection model incorporating word sense information mapped onto upper-level ontologies of differing depths. Based on gold-standard sense information, they achieved large-scale improvements over a basic parse selection model in the context of the Hinoki treebank.

Other notable examples of the successful incorporation of lexical semantics into parsing, not through word sense information but indirectly via selectional preferences, are Dowding et al. (1994) and Hektoen (1997). For a broader review of WSD in NLP applications, see Resnik (2006).

### 3 Integrating Semantics into Parsing

Our approach to providing the parsers with sense information is to make available the semantic denotation of each word in the form of a semantic class. This is done simply by substituting the original words with semantic codes. For example, in the earlier example of *open with a knife* we could substitute both *knife* and *scissors* with the class TOOL, and thus directly facilitate semantic generalisation within the parser. There are three main aspects that we have to consider in this process: (i) the semantic representation, (ii) semantic disambiguation, and (iii) morphology.

There are many ways to represent semantic re-

lationships between words. In this research we opt for a class-based representation that will map semantically-related words into a common semantic category. Our choice for this work was the WordNet 2.1 lexical database, in which synonyms are grouped into synsets, which are then linked via an IS-A hierarchy. WordNet contains other types of relations such as meronymy, but we did not use them in this research. With any lexical semantic resource, we have to be careful to choose the appropriate level of granularity for a given task: if we limit ourselves to synsets we will not be able to capture broader generalisations, such as the one between *knife* and *scissors*;<sup>1</sup> on the other hand by grouping words related at a higher level in the hierarchy we could find that we make overly coarse groupings (e.g. *mallet*, *square* and *steel-wool pad* are also descendants of TOOL in WordNet, none of which would conventionally be used as the manner adjunct of *cut*). We will test different levels of granularity in this work.

The second problem we face is semantic disambiguation. The more fine-grained our semantic representation, the higher the average polysemy and the greater the need to distinguish between these senses. For instance, if we find the word *crane* in a context such as *demolish a house with the crane*, the ability to discern that this corresponds to the DEVICE and not ANIMAL sense of word will allow us to avoid erroneous generalisations. This problem of identifying the correct sense of a word in context is known as word sense disambiguation (WSD: Agirre and Edmonds (2006)). Disambiguating each word relative to its context of use becomes increasingly difficult for fine-grained representations (Palmer et al., 2006). We experiment with different ways of tackling WSD, using both gold-standard data and automatic methods.

Finally, when substituting words with semantic tags we have to decide how to treat different word forms of a given lemma. In the case of English, this pertains most notably to verb inflection and noun number, a distinction which we lose if we opt to map all word forms onto semantic classes. For our current purposes we choose to substitute all word

---

<sup>1</sup>In WordNet 2.1, *knife* and *scissors* are sister synsets, both of which have TOOL as their 4th hypernym. Only by mapping them onto their 1st hypernym or higher would we be able to capture the semantic generalisation alluded to above.

forms, but we plan to look at alternative representations in the future.

## 4 Experimental setting

We evaluate the performance of our approach in two settings: (1) full parsing, and (2) PP attachment within a full parsing context. Below, we outline the dataset used in this research and the parser evaluation methodology, explain the methodology used to perform PP attachment, present the different options for semantic representation, and finally detail the disambiguation methods.

### 4.1 Dataset and parser evaluation

One of the main requirements for our dataset is the availability of gold-standard sense and parse tree annotations. The gold-standard sense annotations allow us to perform upper bound evaluation of the relative impact of a given semantic representation on parsing and PP attachment performance, to contrast with the performance in more realistic semantic disambiguation settings. The gold-standard parse tree annotations are required in order to carry out evaluation of parser and PP attachment performance.

The only publicly-available resource with these two characteristics at the time of this work was the subset of the Brown Corpus that is included in both SemCor (Landes et al., 1998) and the Penn Treebank (PTB).<sup>2</sup> This provided the basis of our dataset. After sentence- and word-aligning the SemCor and PTB data (discarding sentences where there was a difference in tokenisation), we were left with a total of 8,669 sentences containing 151,928 words. Note that this dataset is smaller than the one described by Bikel (2000) in a similar exercise, the reason being our simple and conservative approach taken when merging the resources.

We relied on this dataset alone for all the experiments in this paper. In order to maximise reproducibility and encourage further experimentation in the direction pioneered in this research, we partitioned the data into 3 sets: 80% training, 10% development and 10% test data. This dataset is available on request to the research community.

<sup>2</sup>OntoNotes (Hovy et al., 2006) includes large-scale treebank and (selective) sense data, which we plan to use for future experiments when it becomes fully available.

We evaluate the parsers via labelled bracketing recall ( $\mathcal{R}$ ), precision ( $\mathcal{P}$ ) and F-score ( $\mathcal{F}_1$ ). We use Bikel’s randomized parsing evaluation comparator<sup>3</sup> (with  $p < 0.05$  throughout) to test the statistical significance of the results using word sense information, relative to the respective baseline parser using only lexical features.

### 4.2 PP attachment task

Following Atterer and Schütze (2007), we wrote a script that, given a parse tree, identifies instances of PP attachment ambiguity and outputs the  $(v, n1, p, n2)$  quadruple involved and the attachment decision. This extraction system uses Collins’ rules (based on TREEP (Chiang and Bikel, 2002)) to locate the heads of phrases. Over the combined gold-standard parsing dataset, our script extracted a total of 2,541 PP attachment quadruples. As with the parsing data, we partitioned the data into 3 sets: 80% training, 10% development and 10% test data. Once again, this dataset and the script used to extract the quadruples are available on request to the research community.

In order to evaluate the PP attachment performance of a parser, we run our extraction script over the parser output in the same manner as for the gold-standard data, and compare the extracted quadruples to the gold-standard ones. Note that there is no guarantee of agreement in the quadruple membership between the extraction script and the gold standard, as the parser may have produced a parse which is incompatible with either attachment possibility. A quadruple is deemed correct if: (1) it exists in the gold standard, and (2) the attachment decision is correct. Conversely, it is deemed incorrect if: (1) it exists in the gold standard, and (2) the attachment decision is incorrect. Quadruples not found in the gold standard are discarded. Precision was measured as the number of correct quadruples divided by the total number of correct and incorrect quadruples (i.e. all quadruples which are not discarded), and recall as the number of correct quadruples divided by the total number of gold-standard quadruples in the test set. This evaluation methodology coincides with that of Atterer and Schütze (2007).

Statistical significance was calculated based on

<sup>3</sup>[www.cis.upenn.edu/~dbikel/software.html](http://www.cis.upenn.edu/~dbikel/software.html)

a modified version of the Bikel comparator (see above), once again with  $p < 0.05$ .

### 4.3 Semantic representation

We experimented with a range of semantic representations, all of which are based on WordNet 2.1. As mentioned above, words in WordNet are organised into sets of synonyms, called **synsets**. Each synset in turn belongs to a unique **semantic file** (SF). There are a total of 45 SFs (1 for adverbs, 3 for adjectives, 15 for verbs, and 26 for nouns), based on syntactic and semantic categories. A selection of SFs is presented in Table 1 for illustration purposes.

We experiment with both full synsets and SFs as instances of fine-grained and coarse-grained semantic representation, respectively. As an example of the difference in these two representations, *knife* in its tool sense is in the EDGE TOOL USED AS A CUTTING INSTRUMENT singleton synset, and also in the ARTIFACT SF along with thousands of other words including *cutter*. Note that these are the two extremes of semantic granularity in WordNet, and we plan to experiment with intermediate representation levels in future research (c.f. Li and Abe (1998), McCarthy and Carroll (2003), Xiong et al. (2005), Fujita et al. (2007)).

As a hybrid representation, we tested the effect of merging words with their corresponding SF (e.g. *knife*+ARTIFACT). This is a form of semantic specialisation rather than generalisation, and allows the parser to discriminate between the different senses of each word, but not generalise across words.

For each of these three semantic representations, we experimented with substituting each of: (1) all open-class POSs (nouns, verbs, adjectives and adverbs), (2) nouns only, and (3) verbs only. There are thus a total of 9 combinations of representation type and target POS.

### 4.4 Disambiguation methods

For a given semantic representation, we need some form of WSD to determine the semantics of each token occurrence of a target word. We experimented with three options:

1. **Gold-standard:** Gold-standard annotations from SemCor. This gives us the upper bound performance of the semantic representation.

SF ID	DEFINITION
adj.all	all adjective clusters
adj.pert	relational adjectives (pertainyms)
adj.ppl	participial adjectives
adv.all	all adverbs
noun.act	nouns denoting acts or actions
noun.animal	nouns denoting animals
noun.artifact	nouns denoting man-made objects
...	
verb.consumption	verbs of eating and drinking
verb.emotion	verbs of feeling
verb.perception	verbs of seeing, hearing, feeling
...	

Table 1: A selection of WordNet SFs

2. **First Sense (1ST):** All token instances of a given word are tagged with their most frequent sense in WordNet.<sup>4</sup> Note that the first sense predictions are based largely on the same dataset as we use in our evaluation, such that the predictions are tuned to our dataset and not fully unsupervised.
3. **Automatic Sense Ranking (ASR):** First sense tagging as for First Sense above, except that an unsupervised system is used to automatically predict the most frequent sense for each word based on an independent corpus. The method we use to predict the first sense is that of McCarthy et al. (2004), which was obtained using a thesaurus automatically created from the British National Corpus (BNC) applying the method of Lin (1998), coupled with WordNet-based similarity measures. This method is fully unsupervised and completely unreliant on any annotations from our dataset.

In the case of SFs, we perform full synset WSD based on one of the above options, and then map the prediction onto the corresponding (unique) SF.

## 5 Results

We present the results for each disambiguation approach in turn, analysing the results for parsing and PP attachment separately.

<sup>4</sup>There are some differences with the most frequent sense in SemCor, due to extra corpora used in WordNet development, and also changes in WordNet from the original version used for the SemCor tagging.

SYSTEM	CHARNIAK			BIKEL		
	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$
Baseline	.857	.808	.832	.837	.845	.841
SF	.855	.809	.831	<b>.847*</b>	.854*	.850*
SF <sub>n</sub>	.860	.808	.833	<b>.847*</b>	.853*	.850*
SF <sub>v</sub>	.861	.811	.835	<b>.847*</b>	<b>.856*</b>	<b>.851*</b>
word + SF	<b>.865*</b>	<b>.814*</b>	<b>.839*</b>	.837	.846	.842
word + SF <sub>n</sub>	.862	.809	.835	.841*	.850*	.846*
word + SF <sub>v</sub>	.862	.810	.835	.840	.851	.845
Syn	.863*	.812	.837	.845*	.853*	.849*
Syn <sub>n</sub>	.860	.807	.832	.841	.849	.845
Syn <sub>v</sub>	.863*	.813*	.837*	.843*	.851*	.847*

Table 2: Parsing results with gold-standard senses (\* indicates that the recall or precision is significantly better than baseline; the best performing method in each column is shown in **bold**)

## 5.1 Gold standard

We disambiguated each token instance in our corpus according to the gold-standard sense data, and trained both the Charniak and Bikel parsers over each semantic representation. We evaluated the parsers in full parsing and PP attachment contexts.

The results for parsing are given in Table 2. The rows represent the three semantic representations (including whether we substitute only nouns, only verbs or all POS). We can see that in almost all cases the semantically-enriched representations improve over the baseline parsers. These results are statistically significant in some cases (as indicated by \*). The SF<sub>v</sub> representation produces the best results for Bikel (F-score 0.010 above baseline), while for Charniak the best performance is obtained with word+SF (F-score 0.007 above baseline). Comparing the two baseline parsers, Bikel achieves better precision and Charniak better recall. Overall, Bikel obtains a superior F-score in all configurations.

The results for the PP attachment experiments using gold-standard senses are given in Table 3, both for the Charniak and Bikel parsers. Again, the F-score for the semantic representations is better than the baseline in all cases. We see that the improvement is significant for recall in most cases (particularly when using verbs), but not for precision (only Charniak over Syn<sub>v</sub> and word+SF<sub>v</sub> for Bikel). For both parsers the best results are achieved with SF<sub>v</sub>, which was also the best configuration for parsing with Bikel. The performance gain obtained here is larger than in parsing, which is in accordance with the findings of Stetina and Nagao that lexical semantics has a considerable effect on PP attachment

SYSTEM	CHARNIAK			BIKEL		
	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$
Baseline	.667	.798	.727	.659	.820	.730
SF	.710	.808	.756	.714*	.809	.758
SF <sub>n</sub>	.671	.792	.726	.706	.818	.758
SF <sub>v</sub>	<b>.729*</b>	<b>.823</b>	<b>.773*</b>	<b>.733*</b>	.827	<b>.778*</b>
word + SF	.710*	.801	.753	.706*	<b>.837</b>	.766*
word + SF <sub>n</sub>	.698*	.813	.751	.706*	.829	.763*
word + SF <sub>v</sub>	.714*	.805	.757*	.706*	<b>.837*</b>	.766*
Syn	.722*	.814	.765*	.702*	.825	.758
Syn <sub>n</sub>	.678	.805	.736	.690	.822	.751
Syn <sub>v</sub>	.702*	.817*	.755*	.690*	.834	.755*

Table 3: PP attachment results with gold-standard senses (\* indicates that the recall or precision is significantly better than baseline; the best performing method in each column is shown in **bold**)

performance. As in full-parsing, Bikel outperforms Charniak, but in this case the difference in the baselines is not statistically significant.

## 5.2 First sense (1ST)

For this experiment, we use the first sense data from WordNet for disambiguation. The results for full parsing are given in Table 4. Again, the performance is significantly better than baseline in most cases, and surprisingly the results are even better than gold-standard in some cases. We hypothesise that this is due to the avoidance of excessive fragmentation, as occurs with fine-grained senses. The results are significantly better for nouns, with SF<sub>n</sub> performing best. Verbs seem to suffer from lack of disambiguation precision, especially for Bikel. Here again, Charniak trails behind Bikel.

The results for the PP attachment task are shown in Table 5. The behaviour is slightly different here, with Charniak obtaining better results than Bikel in most cases. As was the case for parsing, the performance with 1ST reaches and in many instances surpasses gold-standard levels, achieving statistical significance over the baseline in places. Comparing the semantic representations, the best results are achieved with SF<sub>v</sub>, as we saw in the gold-standard PP-attachment case.

## 5.3 Automatic sense ranking (ASR)

The final option for WSD is automatic sense ranking, which indicates how well our method performs in a completely unsupervised setting.

The parsing results are given in Table 6. We can see that the scores are very similar to those from

SYSTEM	CHARNIAK			BIKEL		
	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$
Baseline	.857	.807	.832	.837	.845	.841
SF	.851	.804	.827	.843	.850	.846
SF <sub>n</sub>	<b>.863*</b>	<b>.813</b>	<b>.837*</b>	<b>.850*</b>	<b>.854*</b>	<b>.852*</b>
SF <sub>v</sub>	.857	.808	.832	.843	.853*	.848
word + SF	.859	.810	.834	.833	.841	.837
word + SF <sub>n</sub>	.862*	.811	.836	.844*	.851*	.848*
word + SF <sub>v</sub>	.857	.808	.832	.831	.839	.835
Syn	.857	.810	.833	.837	.844	.840
Syn <sub>n</sub>	<b>.863*</b>	.812	<b>.837*</b>	.844*	.851*	.848*
Syn <sub>v</sub>	.860	.810	.834	.836	.844	.840

Table 4: Parsing results with 1ST (\* indicates that the recall or precision is significantly better than baseline; the best performing method in each column is shown in **bold**)

SYSTEM	CHARNIAK			BIKEL		
	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$
Baseline	.667	.798	.727	.659	.820	.730
SF	.710	.808	.756	.702	.806	.751
SF <sub>n</sub>	.671	.781	.722	.702	<b>.829</b>	.760
SF <sub>v</sub>	<b>.737*</b>	<b>.836*</b>	<b>.783*</b>	<b>.718*</b>	.821	<b>.766*</b>
word + SF	.706	.811	.755	.694	.823	.753
word + SF <sub>n</sub>	.690	.815	.747	.667	.810	.731
word + SF <sub>v</sub>	.714*	.805	.757*	.710*	.819	.761*
Syn	.725*	.833*	.776*	.698	.828	.757
Syn <sub>n</sub>	.698	.828*	.757*	.667	.817	.734
Syn <sub>v</sub>	.722*	.811	.763*	.706*	.818	.758*

Table 5: PP attachment results with 1ST (\* indicates that the recall or precision is significantly better than baseline; the best performing method in each column is shown in **bold**)

1ST, with improvements in some cases, particularly for Charniak. Again, the results are better for nouns, except for the case of SF<sub>v</sub> with Bikel. Bikel outperforms Charniak in terms of F-score in all cases.

The PP attachment results are given in Table 7. The results are similar to 1ST, with significant improvements for verbs. In this case, synsets slightly outperform SF. Charniak performs better than Bikel, and the results for Syn<sub>v</sub> are higher than the best obtained using gold-standard senses.

## 6 Discussion

The results of the previous section show that the improvements in parsing results are small but significant, for all three word sense disambiguation strategies (gold-standard, 1ST and ASR). Table 8 summarises the results, showing that the error reduction rate (ERR) over the parsing F-score is up to 6.9%, which is remarkable given the relatively superficial strategy for incorporating sense information into the parser. Note also that our baseline results for the

SYSTEM	CHARNIAK			BIKEL		
	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$
Baseline	.857	.807	.832	.837	.845	.841
SF	.863	<b>.815*</b>	<b>.838</b>	.845*	.852	.849
SF <sub>n</sub>	.862	.810	.835	.845*	.850	.847*
SF <sub>v</sub>	.859	.810	.833	<b>.846*</b>	<b>.856*</b>	<b>.851*</b>
word + SF	.859	.810	.834	.836	.844	.840
word + SF <sub>n</sub>	<b>.865*</b>	.813*	<b>.838*</b>	.844*	.852*	.848*
word + SF <sub>v</sub>	.856	.806	.830	.832	.839	.836
Syn	.856	.807	.831	.840	.847	.843
Syn <sub>n</sub>	.864*	.813*	<b>.838*</b>	.844*	.851*	.847*
Syn <sub>v</sub>	.857	.806	.831	.837	.845	.841

Table 6: Parsing results with ASR (\* indicates that the recall or precision is significantly better than baseline; the best performing method in each column is shown in **bold**)

SYSTEM	CHARNIAK			BIKEL		
	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{F}_1$
Baseline	.667	.798	.727	.659	.820	.730
SF	<b>.733*</b>	.824	.776*	.698	.805	.748
SF <sub>n</sub>	.682	.791	.733	.671	.807	.732
SF <sub>v</sub>	<b>.733*</b>	.813	.771*	.710*	.812	.757*
word + SF	.714*	.798	.754	.675	.800	.732
word + SF <sub>n</sub>	.690	.807	.744	.659	.804	.724
word + SF <sub>v</sub>	.706*	.800	.750	.702*	.814	.754*
Syn	<b>.733*</b>	<b>.827</b>	<b>.778*</b>	.694	.805	.745
Syn <sub>n</sub>	.686	.810	.743	.667	.806	.730
Syn <sub>v</sub>	.714*	.816	.762*	<b>.714*</b>	<b>.816</b>	<b>.762*</b>

Table 7: PP attachment results with ASR (\* indicates that the recall or precision is significantly better than baseline; the best performance in each column is shown in **bold**)

dataset are almost the same as previous work parsing the Brown corpus with similar models (Gildea, 2001), which suggests that our dataset is representative of this corpus.

The improvement in PP attachment was larger (20.5% ERR), and also statistically significant. The results for PP attachment are especially important, as we demonstrate that the sense information has high utility when embedded within a parser, where the parser needs to first identify the ambiguity and heads correctly. Note that Atterer and Schütze (2007) have shown that the Bikel parser performs as well as the state-of-the-art in PP attachment, which suggests our method improves over the current state-of-the-art. The fact that the improvement is larger for PP attachment than for full parsing is suggestive of PP attachment being a parsing subtask where lexical semantic information is particularly important, supporting the findings of Stetina and Nagao (1997) over a standalone PP attachment task. We also observed that while better PP-attachment usually improves parsing, there is some small variation. This

WSD	TASK	PAR	BASE	SEM	ERR	BEST	
Gold-standard	Pars.	C	.832	.839*	4.2%	word+SF	
		B	.841	.851*	6.3%	SF <sub>v</sub>	
		C	.727	.773*	16.9%	SF <sub>v</sub>	
	PP	B	.730	.778*	17.8%	SF <sub>v</sub>	
		Pars.	C	.832	.837*	3.0%	SF <sub>n</sub> , Syn <sub>n</sub>
			B	.841	.852*	6.9%	SF <sub>n</sub>
C	.727		.783*	20.5%	SF <sub>v</sub>		
1ST	PP	B	.730	.766*	13.3%	SF <sub>v</sub>	
		Pars.	C	.832	.838*	3.6%	SF, word+SF <sub>n</sub> , Syn <sub>n</sub>
			B	.841	.851*	6.3%	SF <sub>v</sub>
	C		.727	.778*	18.7%	Syn	
	ASR	PP	B	.730	.762*	11.9%	Syn <sub>v</sub>

Table 8: Summary of F-score results with error reduction rates and the best semantic representation(s) for each setting (C = Charniak, B = Bikel)

means that the best configuration for PP-attachment does not always produce the best results for parsing

One surprising finding was the strong performance of the automatic WSD systems, actually outperforming the gold-standard annotation overall. Our interpretation of this result is that the approach of annotating all occurrences of the same word with the same sense allows the model to avoid the data sparseness associated with the gold-standard distinctions, as well as supporting the merging of different words into single semantic classes. While the results for gold-standard senses were intended as an upper bound for WordNet-based sense information, in practice there was very little difference between gold-standard senses and automatic WSD in all cases barring the Bikel parser and PP attachment.

Comparing the two parsers, Charniak performs better than Bikel on PP attachment when automatic WSD is used, while Bikel performs better on parsing overall. Regarding the choice of WSD system, the results for both approaches are very similar, showing that ASR performs well, even if it does not require sense frequency information.

The analysis of performance according to the semantic representation is not so clear cut. Generalising only verbs to semantic files (SF<sub>v</sub>) was the best option in most of the experiments, particularly for PP-attachment. This could indicate that semantic generalisation is particularly important for verbs, more so than nouns.

Our hope is that this paper serves as the bridge-head for a new line of research into the impact of lexical semantics on parsing. Notably, more could be done to fine-tune the semantic representation be-

tween the two extremes of full synsets and SFs. One could also imagine that the appropriate level of generalisation differs across POS and even the relative syntactic role, e.g. finer-grained semantics are needed for the objects than subjects of verbs.

On the other hand, the parsing strategy is very simple, as we just substitute words by their semantic class and then train statistical parsers on the transformed input. The semantic class should be an information source that the parsers take into account in addition to analysing the actual words used. Tighter integration of semantics into the parsing models, possibly in the form of discriminative reranking models (Collins and Koo, 2005; Charniak and Johnson, 2005; McClosky et al., 2006), is a promising way forward in this regard.

## 7 Conclusions

In this work we have trained two state-of-the-art statistical parsers on semantically-enriched input, where content words have been substituted with their semantic classes. This simple method allows us to incorporate lexical semantic information into the parser, without having to reimplement a full statistical parser. We tested the two parsers in both a full parsing and a PP attachment context.

This paper shows that semantic classes achieve significant improvement both on full parsing and PP attachment tasks relative to the baseline parsers. PP attachment achieves a 20.5% ERR, and parsing 6.9% without requiring hand-tagged data.

The results are highly significant in demonstrating that a simplistic approach to incorporating lexical semantics into a parser significantly improves parser performance. As far as we know, these are the first results over both WordNet and the Penn Treebank to show that semantic processing helps parsing.

## Acknowledgements

We wish to thank Diana McCarthy for providing us with the sense rank for the target words. This work was partially funded by the Education Ministry (project KNOW TIN2006-15049), the Basque Government (IT-397-07), and the Australian Research Council (grant no. DP0663879). Eneko Agirre participated in this research while visiting the University of Melbourne, based on joint funding from the Basque Government and HCSNet.



## References

- Eneko Agirre and Philip Edmonds, editors. 2006. *Word Sense Disambiguation: Algorithms and Applications*. Springer, Dordrecht, Netherlands.
- Michaela Atterer and Hinrich Schütze. 2007. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476.
- Daniel M. Bikel. 2000. A statistical model for parsing and word-sense disambiguation. In *Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 155–63, Hong Kong, China.
- Daniel M. Bikel. 2004. Intricacies of Collins’ parsing model. *Computational Linguistics*, 30(4):479–511.
- Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of the 43rd Annual Meeting of the ACL*, pages 173–80, Ann Arbor, USA.
- Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. of the 15th Annual Conference on Artificial Intelligence (AAAI-97)*, pages 598–603, Stanford, USA.
- Eugene Charniak. 2000. A maximum entropy-based parser. In *Proc. of the 1st Annual Meeting of the North American Chapter of Association for Computational Linguistics (NAACL2000)*, Seattle, USA.
- David Chiang and David M. Bikel. 2002. Recovering latent information in treebanks. In *Proc. of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 183–9, Taipei, Taiwan.
- Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.
- Michael J. Collins. 1996. A new statistical parser based on lexical dependencies. In *Proc. of the 34th Annual Meeting of the ACL*, pages 184–91, Santa Cruz, USA.
- Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.
- John Dowding, Robert Moore, François Andry, and Douglas Moran. 1994. Interleaving syntax and semantics in an efficient bottom-up parser. In *Proc. of the 32nd Annual Meeting of the ACL*, pages 110–6, Las Cruces, USA.
- Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.
- Sanae Fujita, Francis Bond, Stephan Oepen, and Takaaki Tanaka. 2007. Exploiting semantic information for HPSG parse selection. In *Proc. of the ACL 2007 Workshop on Deep Linguistic Processing*, pages 25–32, Prague, Czech Republic.
- Daniel Gildea. 2001. Corpus variation and parser performance. In *Proc. of the 6th Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, pages 167–202, Pittsburgh, USA.
- Erik Hektoen. 1997. Probabilistic parse selection based on semantic cooccurrences. In *Proc. of the 5th International Workshop on Parsing Technologies (IWPT-1997)*, pages 113–122, Boston, USA.
- Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proc. of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA.
- Shari Landes, Claudia Leacock, and Randee I. Tengi. 1998. Building semantic concordances. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.
- Hang Li and Naoki Abe. 1998. Generalising case frames using a thesaurus and the MDL principle. *Computational Linguistics*, 24(2):217–44.
- Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of the 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics: COLING/ACL-98*, pages 768–774, Montreal, Canada.
- David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proc. of the 33rd Annual Meeting of the ACL*, pages 276–83, Cambridge, USA.
- 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.
- Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.
- Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant senses in untagged text. In *Proc. of the 42nd Annual Meeting of the ACL*, pages 280–7, Barcelona, Spain.
- David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proc. of the Human Language Technology Conference of the NAACL (NAACL2006)*, pages 152–159, New York City, USA.
- Brian Mitchell. 2004. *Prepositional Phrase Attachment using Machine Learning Algorithms*. Ph.D. thesis, University of Sheffield.
- Martha Palmer, Hoa Dang, and Christiane Fellbaum. 2006. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, 13(2):137–63.
- Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *HLT ’94: Proceedings of the Workshop on Human Language Technology*, pages 250–255, Plainsboro, USA.
- Philip Resnik. 2006. WSD in NLP applications. In Eneko Agirre and Philip Edmonds, editors, *Word Sense Disambiguation: Algorithms and Applications*, chapter 11, pages 303–40. Springer, Dordrecht, Netherlands.
- Jiri Stetina and Makoto Nagao. 1997. Corpus based PP attachment ambiguity resolution with a semantic dictionary. In *Proc. of the 5th Annual Workshop on Very Large Corpora*, pages 66–80, Hong Kong, China.
- Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin, and Yueliang Qian. 2005. Parsing the Penn Chinese Treebank with semantic knowledge. In *Proc. of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*, pages 70–81, Jeju Island, Korea.

# A Logical Basis for the D Combinator and Normal Form in CCG

Frederick Hoyt and Jason Baldridge

The Department of Linguistics

The University of Texas at Austin

{fmhoyt, jbaldrid}@mail.utexas.edu

## Abstract

The standard set of rules defined in Combinatory Categorical Grammar (CCG) fails to provide satisfactory analyses for a number of syntactic structures found in natural languages. These structures can be analyzed elegantly by augmenting CCG with a class of rules based on the combinator **D** (Curry and Feys, 1958). We show two ways to derive the **D** rules: one based on unary composition and the other based on a logical characterization of CCG’s rule base (Baldridge, 2002). We also show how Eisner’s (1996) normal form constraints follow from this logic, ensuring that the **D** rules do not lead to spurious ambiguities.

## 1 Introduction

Combinatory Categorical Grammar (CCG, Steedman (2000)) is a compositional, semantically transparent formalism that is both linguistically expressive and computationally tractable. It has been used for a variety of tasks, such as wide-coverage parsing (Hockenmaier and Steedman, 2002; Clark and Curran, 2007), sentence realization (White, 2006), learning semantic parsers (Zettlemoyer and Collins, 2007), dialog systems (Kruijff et al., 2007), grammar engineering (Beavers, 2004; Baldridge et al., 2007), and modeling syntactic priming (Reitter et al., 2006).

A distinctive aspect of CCG is that it provides a very flexible notion of constituency. This supports elegant analyses of several phenomena (e.g., coordination, long-distance extraction, and intonation) and allows incremental parsing with the competence grammar (Steedman, 2000). Here, we argue

that even with its flexibility, CCG as standardly defined is not permissive enough for certain linguistic constructions and greater incrementality. Following Wittenburg (1987), we remedy this by adding a set of rules based on the **D** combinator of combinatory logic (Curry and Feys, 1958).

$$(1) \quad x/(y/z):f \quad y/w:g \Rightarrow x/(w/z):\lambda h.f(\lambda x.g h x)$$

We show that CCG augmented with this rule improves CCG’s empirical coverage by allowing better analyses of modal verbs in English and causatives in Spanish, and certain coordinate constructions.

The **D** rules are well-behaved; we show this by deriving them both from unary composition and from the logic defined by Baldridge (2002). Both perspectives on **D** ensure that the new rules are compatible with normal form constraints (Eisner, 1996) for controlling spurious ambiguity. The logic also ensures that the new rules are subject to modalities consistent with those defined by Baldridge and Kruijff (2003). Furthermore, we define a logic that produces Eisner’s constraints as grammar internal theorems rather than parsing stipulations.

## 2 Combinatory Categorical Grammar

CCG uses a universal set of syntactic rules based on the **B**, **T**, and **S** combinators of combinatory logic (Curry and Feys, 1958):

$$(2) \quad \mathbf{B}: ((\mathbf{B}f)g)x = f(gx)$$

$$\mathbf{T}: \mathbf{T}x f = f x$$

$$\mathbf{S}: ((\mathbf{S}f)g)x = f x(gx)$$

CCG functors are functions over strings of symbols, so different linearized versions of each of the combinators have to be specified (ignoring **S** here):

- (3) **FA:** ( $>$ )  $x/_*y \ y \Rightarrow x$   
 ( $<$ )  $y \ x\_*y \Rightarrow x$
- B:** ( $>\mathbf{B}$ )  $x/_\diamond y \ y/_\diamond z \Rightarrow x/_\diamond z$   
 ( $<\mathbf{B}$ )  $y\_ \diamond z \ x\_ \diamond y \Rightarrow x\_ \diamond z$   
 ( $>\mathbf{B}_\times$ )  $x/_\times y \ y\_ \times z \Rightarrow x\_ \times z$   
 ( $<\mathbf{B}_\times$ )  $y\_ \times z \ x\_ \times y \Rightarrow x\_ \times z$
- T:** ( $>\mathbf{T}$ )  $x \Rightarrow t/_i(t\_i x)$   
 ( $<\mathbf{T}$ )  $x \Rightarrow t\_i(t/_i x)$

The symbols  $\{*, \diamond, \times, \cdot\}$  are modalities that allow subtypes of slashes to be defined; this in turn allows the slashes on categories to be defined in a way that allows them to be used (or not) with specific subsets of the above rules. The rules of this multimodal version of CCG (Baldrige, 2002; Baldrige and Kruijff, 2003) are derived as theorems of a Categorical Type Logic (CTL, Moortgat (1997)).

This treats CCG as a compilation of CTL proofs, providing a principled, grammar-internal basis for restrictions on the CCG rules, transferring language-particular restrictions on rule application to the lexicon, and allowing the CCG rules to be viewed as grammatical universals (Baldrige and Kruijff, 2003; Steedman and Baldrige, To Appear).

These rules—especially the **B** rules—allow derivations to be *partially* associative: given appropriate type assignments, a string ABC can be analyzed as either A(BC) or (AB)C. This associativity leads to elegant analyses of phenomena that demand more effort in less flexible frameworks. One of the best known is “odd constituent” coordination:

- (4) Bob gave **Stan a beer** and **Max a coke**.  
 (5) **I will buy** and **you will eat** a cheeseburger.

The coordinated constituents are challenging because they are at odds with standardly assumed phrase structure constituents. In CCG, such constituents simply follow from the associativity added by the **B** and **T** rules. For example, given the category assignments in (6) and the abbreviations in (7), (4) is analyzed as in (8) and (9). Each conjunct is a pair of type-raised NPs combined by means of the  $>\mathbf{B}$ -rule, deriving two composed constituents that are arguments to the conjunction:<sup>1</sup>

- (6) i. **Bob**  $\vdash s/(s\_np)$

<sup>1</sup>We follow (Steedman, 2000) in assuming that type-raising applies in the lexicon, and therefore that nominals such as *Stan*

- ii. **Stan, Max**  $\vdash$   
 $((s\_np)/np)\(((s\_np)/np)/np)$
- iii. **a beer, a coke**  $\vdash (s\_np)\((s\_np)/np)$
- iv. **and**  $\vdash (x\_x)/_x x$
- v. **gave**  $\vdash ((s\_np)/np)/np$
- (7) i.  $vp = s\_np$   
 ii.  $tv = (s\_np)/np$   
 iii.  $dtv = ((s\_np)/np)/np$

(8) **Stan a beer and Max a coke**

$$\frac{\frac{tv\_dt \quad vp\_tv}{vp\_dt} \leftarrow \mathbf{B} \quad \frac{(x\_x)/_x x \quad tv\_dt \quad vp\_tv}{vp\_dt} \leftarrow \mathbf{B}}{\frac{(vp\_dt)\(vp\_dt)}{vp\_dt} \rightarrow}$$

(9) **Bill gave Stan a beer and Max a coke**

$$\frac{\frac{s\_vp \quad dt \quad vp\_dt}{vp} \leftarrow \quad \frac{vp\_dt}{s} \rightarrow}{s}$$

Similarly, *I will buy* is derived with category  $s\_np$  by assuming the category (6i) for *I* and composing that with both verbs in turn.

CCG’s approach is appealing because such constituents are not odd at all: they simply follow from the fact that CCG is a system of type-based grammatical inference that allows left associativity.

### 3 Linguistic Motivation for D

CCG is only *partially* associative. Here, we discuss several situations which require greater associativity and thus cannot be given an adequate analysis with CCG as standardly defined. These structures have in common that a category of the form  $x|(y|z)$  must combine with one of the form  $y|w$ —exactly the configuration handled by the **D** schemata in (1).

#### 3.1 Cross-Conjunct Extraction

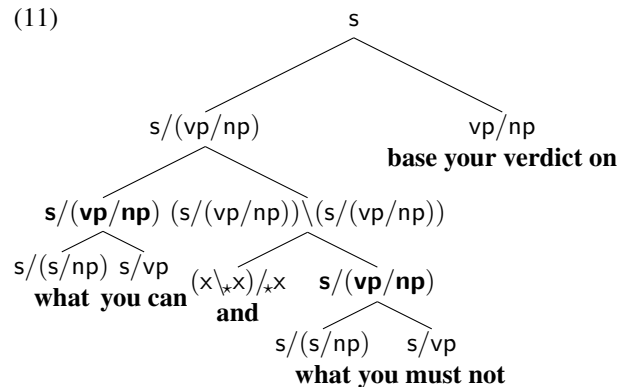
In the first situation, a question word is distributed across auxiliary or subordinating verb categories:

- (10) ... **what you can** and **what you must not** base your verdict on.

We call this *cross-conjunct extraction*. It was noted by Pickering and Barry (1993) for English, but to the best of our knowledge it has not been treated in the

have type-raised lexical assignments. We also suppress semantic representations in the derivations for the sake of space.

CCG literature, nor noted in other languages. The problem it presents to CCG is clear in (11), which shows the necessary derivation of (10) using standard multimodal category assignments. For the tokens of *what* to form constituents with *you can* and *you must not*, they must combine directly. The problem is that these constituents (in bold) cannot be created with the standard CCG combinators in (3).



The category for *and* is marked for non-associativity with  $*$ , and thus combines with other expressions only by function application (Baldrige, 2002). This ensures that each conjunct is a discrete constituent.

Cross-conjunct extraction occurs in other languages as well, including Dutch (12), German (13), Romanian (14), and Spanish (15):

- (12) *dat ik haar wil en dat ik haar moet helpen.*  
that I her want and that I her can help  
“... that I want to and that I can help her.”
- (13) *Wen kann ich und wen darf ich noch wählen?*  
who can I and who may I still choose  
“Whom can I and whom may I still chose?”
- (14) *Gandeste-te cui ce vrei,*  
consider.imper.2s-refl.2s who.dat what want.2s  
*și cui ce poți, să dai.*  
and who.dat what can.2s to give.subj.2s  
“Consider to whom you want and to whom you are able to give what.”
- (15) *Me lo puedes y me lo debes explicar*  
me it can.2s and me it must.2s ask  
“You can and should explain it to me.”

It is thus a general phenomenon, not just a quirk of English. While it could be handled with extra categories, such as  $(s/(vp/np))/(s/np)$  for *what*, this is exactly the sort of strong-arm tactic that inclusion of the standard **B**, **T**, and **S** rules is meant to avoid.

### 3.2 English Auxiliary Verbs

The standard CCG analysis for English auxiliary verbs is the type exemplified in (16) (Steedman, 2000, 68), interpreted as a unary operator over sentence meanings (Gamut, 1991; Kratzer, 1991):

$$(16) \text{ can} \vdash (s \backslash np) / (s \backslash np) : \lambda P_{et} \lambda x. \diamond P(x)$$

However, this type is empirically underdetermined, given a widely-noted set of generalizations suggesting that auxiliaries and raising verbs take no subject argument at all (Jacobson, 1990, a.o.).

- (17) i. Lack of syntactic restrictions on the subject;  
ii. Lack of semantic restrictions on the subject;  
iii. Inheritance of selectional restrictions from the subordinate predicate.

Two arguments are made for (16). First, it is necessary so that type-raised subjects can compose with the auxiliary in extraction contexts, as in (18):

$$(18) \begin{array}{cccc} \text{what} & \text{I} & \text{can} & \text{eat} \\ \hline s/(s/np) & s/vp & vp/vp & tv \\ \hline & & \xrightarrow{B} & \\ & & s/vp & \\ \hline & & & \xrightarrow{B} \\ & & & s/np \\ \hline & & & \xrightarrow{S} \\ & & & s \end{array}$$

Second, it is claimed to be necessary in order to account for subject-verb agreement, on the assumption that agreement features are domain restrictions on functors of type  $s \backslash np$  (Steedman, 1992, 1996).

The first argument is the topic of this paper, and, as we show below, is refuted by the use of the **D**-combinator. The second argument is undermined by examples like (19):

- (19) There **appear** to have been [ neither [ **any catastrophic consequences** ], nor [ **a drastic change** in the average age of retirement ] ] .

In (19), *appear* agrees with two negative-polarity-sensitive NPs trapped inside a *neither-nor* coordinate structure in which they are licensed. *Appear* therefore does not combine with them directly, showing that the agreement relation need not be mediated by direct application of a subject argument.

We conclude, therefore, that the assignment of the  $vp/vp$  type to English auxiliaries and modal verbs is unsupported on both formal and linguistic grounds.

Following Jacobson (1990), a more empirically-motivated assignment is (20):

(20)  $\text{can} \vdash s/s : \lambda p_t. \diamond p$

Combining (20) with a type-raised subject presents another instance of the structure in (1), where that question words are represented as variable-binding operators (Groenendijk and Stokhof, 1997):

(21) 
$$\frac{\text{what} \quad \text{I} \quad \text{can}}{s/(s/np) : \lambda Q_{et} ?yQy \quad \frac{s/vp : \lambda P_{et}. P'i'}{***} \quad \frac{s/s : \lambda p_t. \diamond p}{***}}{>B ***}$$

### 3.3 The Spanish Causative Construction

The schema in (1) is also found in the widely-studied Romance causative construction (Andrews and Manning, 1999, a.m.o), illustrated in (22):

(22) *Nos hizo leer El Señor de los Anillos.*  
cl.1p made.3s read the Lord of the Rings  
“He made us read *The Lord of the Rings*.”

The aspect of the construction that is relevant here is that the causative verb *hacer* appears to take an object argument understood as the subject or agent of the subordinate verb (the *causee*). However, it has been argued that Spanish causative verbs do not in fact take objects (Ackerman and Moore, 1999, and refs therein). There are two arguments for this.

First, syntactic alternations that apply to object-taking verbs, such as passivization and periphrasis with subjunctive complements, do not apply to *hacer* (Luján, 1980). Second, *hacer* specifies neither the case form of the *causee*, nor any semantic entailments with respect to it. These are instead determined by syntactic, semantic, and pragmatic factors, such as transitivity, word order, animacy, gender, social prestige, and referential specificity (Finnemann, 1982, a.o). Thus, there is neither syntactic nor semantic evidence that *hacer* takes an object argument.

On this basis, we assign *hacer* the category (23):

(23)  $\text{hacer} \vdash (s \setminus np)/s : \lambda P \lambda x. \text{cause}' P x$

However, Spanish has examples of cross-conjunct extraction in which *hacer* hosts clitics:

(24) *No solo le ordenaron, sino que*  
not only cl.dat.3ms ordered.3p but  
*le hicieron barrer la verada.*  
cl.dat.3ms made.3p sweep the sidewalk  
“They not only ordered him to, but also made him sweep the sidewalk.”

This shows another instance of the schema in (1), which is undefined for any of the combinators in (3):

(25) 
$$\frac{\text{le} \quad \text{hicieron} \quad \text{barrer la verada}}{\frac{(s \setminus np)/((s \setminus np)/np) \quad (s \setminus np)/s}{***} \quad (s \setminus np)}{>B ***}$$

### 3.4 Analyses Based on D

The preceding data motivates adding **D** rules (we return to the distribution of the modalities below):

(26) 
$$\begin{aligned} >D & \quad x/o(y/o/z) \quad y/o/w \Rightarrow x/o(w/o/z) \\ >D_x & \quad x/x(y/x/z) \quad y \setminus_x w \Rightarrow x \setminus_x (w/x/z) \\ >D_{ox} & \quad x/o(y \setminus_x z) \quad y/w \Rightarrow x/o(w \setminus_x z) \\ >D_{xo} & \quad x/x(y \setminus_o z) \quad y \setminus_w \Rightarrow x \setminus_x (w \setminus_o z) \end{aligned}$$

(27) 
$$\begin{aligned} <D & \quad y \setminus_o w \quad x \setminus_o (y \setminus_o z) \Rightarrow x \setminus_o (w \setminus_o z) \\ <D_x & \quad y/xw \quad x \setminus_x (y \setminus_x z) \Rightarrow x \setminus_x (w \setminus_x z) \\ <D_{ox} & \quad y \setminus_w \quad x \setminus_o (y/xz) \Rightarrow x \setminus_o (w/xz) \\ <D_{xo} & \quad y/w \quad x \setminus_x (y/o/z) \Rightarrow x \setminus_x (w/o/z) \end{aligned}$$

To illustrate with example (10), one application of  $>D$  allows *you* and *can* to combine when the auxiliary is given the principled type assignment  $s/s$ , and another combines *what* with the result.

(28) 
$$\frac{\text{what} \quad \text{you} \quad \text{can}}{\frac{s/o(s/np) \quad \frac{s/o(s \setminus_x np) \quad s/s}{>D_{ox}}}{s/o(s \setminus_x np)}}{s/o((s \setminus_x np)/np)}{>D}$$

The derivation then proceeds in the usual way.

Likewise, **D** handles the Spanish causative constructions (29) straightforwardly :

(29) 
$$\frac{\text{lo} \quad \text{hice} \quad \text{dormir}}{\frac{(s \setminus np)/o((s \setminus np)/o np) \quad (s \setminus np)/o s \quad s/np}{>D}}{s \setminus np}$$

The **D**-rules thus provide straightforward analyses of such constructions by delivering flexible constituency while maintaining CCG’s commitment to low categorial ambiguity and semantic transparency.

## 4 Deriving Eisner Normal Form

Adding new rules can have implications for parsing efficiency. In this section, we show that the **D** rules fit naturally within standard normal form constraints for CCG parsing (Eisner, 1996), by providing both

combinatory and logical bases for **D**. This additionally allows Eisner’s normal form constraints to be derived as grammar internal theorems.

#### 4.1 The Spurious Ambiguity Problem

CCG’s flexibility is useful for linguistic analyses, but leads to *spurious ambiguity* (Wittenburg, 1987) due to the associativity introduced by the **B** and **T** rules. This can incur a high computational cost which parsers must deal with. Several techniques have been proposed for the problem (Wittenburg, 1987; Karttunen, 1989; Hepple and Morrill, 1989; Eisner, 1996). The most commonly used are Karttunen’s chart subsumption check (White and Baldrige, 2003; Hockenmaier and Steedman, 2002) and Eisner’s normal-form constraints (Bozsahin, 1998; Clark and Curran, 2007).

Eisner’s normal form, referred to here as *Eisner NF* and paraphrased in (30), has the advantage of not requiring comparisons of logical forms: it functions purely on the syntactic types being combined.

- (30) For a set  $S$  of semantically equivalent<sup>2</sup> parse trees for a string  $ABC$ , admit the unique parse tree such that at least one of (i) or (ii) holds:
- i.  $C$  is not the argument of  $(AB)$  resulting from application of  $>\mathbf{B}^{1+}$ .
  - ii.  $A$  is not the argument of  $(BC)$  resulting from application of  $<\mathbf{B}^{1+}$ .

The implication is that outputs of  $\mathbf{B}^{1+}$  rules are *inert*, using the terminology of Baldrige (2002). Inert slashes are Baldrige’s (2002) encoding in OpenCCG<sup>3</sup> of his CTL interpretation of Steedman’s (2000) *antecedent-government* feature.

Eisner derives (30) from two theorems about the set of semantically equivalent parses that a CCG parser will generate for a given string (see (Eisner, 1996) for proofs and discussion of the theorems):

- (31) *Theorem 1*: For every parse tree  $\alpha$ , there is a semantically equivalent parse-tree  $NF(\alpha)$  in which no node resulting from application of **B** or **S** functions as the primary functor in a rule application.
- (32) *Theorem 2*: If  $NF(\alpha)$  and  $NF(\alpha')$  are distinct parse trees, then their model-theoretic interpretations are distinct.

<sup>2</sup>Two parse trees are semantically equivalent if: (i) their leaf nodes have equivalent interpretations, and (ii) equivalent scope relations hold between their respective leaf-node meanings.

<sup>3</sup><http://openccg.sourceforge.net>

Eisner uses a generalized form  $\mathbf{B}^n$  ( $n \geq 0$ ) of composition that subsumes function application.<sup>4</sup>

$$(33) \quad >\mathbf{B}^n: x/y \quad y\$^n \Rightarrow x\$^n$$

$$(34) \quad <\mathbf{B}^n: y\$^n \quad x \backslash y \Rightarrow x\$^n$$

Based on these theorems, Eisner defines NF as follows (for  $R, S, T$  as  $\mathbf{B}^n$  or **S**, and  $Q = \mathbf{B}^{n \geq 1}$ ):

- (35) Given a parse tree  $\alpha$ :
- i. If  $\alpha$  is a lexical item, then  $\alpha$  is in Eisner-NF.
  - ii. If  $\alpha$  is a parse tree  $\langle R, \beta, \gamma \rangle$  and  $NF(\beta)$ ,  $NF(\gamma)$ , then  $NF(\alpha)$ .
  - iii. If  $\beta$  is not in Eisner-NF, then  $NF(\beta) = \langle Q, \beta_1, \beta_2 \rangle$ , and  $NF(\alpha) = \langle S, \beta_1, NF(\langle T, \beta_2, \gamma \rangle) \rangle$ .

As a parsing constraint, (30) is a filter on the set of parses produced for a given string. It preserves all the unique semantic forms generated for the string while eliminating all spurious ambiguities: it is both *safe* and *complete*.

Given the utility of Eisner NF for practical CCG parsing, the **D** rules we propose should be compatible with (30). This requires that the generalizations underlying (30) apply to **D** as well. In the remainder of this section, we show this in two ways.

#### 4.2 Deriving D from B

The first is to derive the binary **B** rules from a unary rule based on the unary combinator  $\hat{\mathbf{B}}$ :<sup>5</sup>

$$(36) \quad x/y : f_{xy} \Rightarrow (x/z)/(y/z) : \lambda h_{zy} \lambda x_z. f(hx)$$

We then derive **D** from  $\hat{\mathbf{B}}$  and show that clause (iii) of (35) holds of  $Q$  schematized over both **B** and **D**.

Applying **D** to an argument sequence is equivalent to compound application of binary **B**:

$$(37) \quad (((\mathbf{D}f)g)h)x = (fg)(hx)$$

$$(38) \quad (((\mathbf{B}\mathbf{B})f)g)h)x = ((\mathbf{B}(fg))h)x = (fg)(hx)$$

Syntactically, binary **B** is equivalent to application of unary  $\hat{\mathbf{B}}$  to the primary functor  $\Delta$ , followed by applying the secondary functor  $\Gamma$  to the output of  $\hat{\mathbf{B}}$  by means of function application (Jacobson, 1999):

<sup>4</sup>We use Steedman’s (Steedman, 1996) “\$”-convention for representing argument stacks of length  $n$ , for  $n \geq 0$ .

<sup>5</sup>This is Lambek’s (1958) *Division* rule, also known as the “*Geach* rule” (Jacobson, 1999).

$$(39) \quad \frac{\frac{\Delta}{x/y} \quad \frac{\Gamma}{y/z}}{\frac{(x/z)/(y/z)}{x/z} \hat{\mathbf{B}}} \hat{\mathbf{B}}$$

$\mathbf{B}^n$  ( $n \geq 1$ ) is derived by applying  $\hat{\mathbf{B}}$  to the primary functor  $n$  times. For example,  $\mathbf{B}^2$  is derived by 2 applications of  $\hat{\mathbf{B}}$  to the primary functor:

$$(40) \quad \frac{\frac{\frac{\Delta}{x/y} \quad \frac{\Gamma}{(y/w)/z}}{(x/w)/(y/w)} \hat{\mathbf{B}}}{\frac{((x/w)/z)/((y/w)/z)}{(x/w)/z} \hat{\mathbf{B}}} \hat{\mathbf{B}}$$

The rules for  $\mathbf{D}$  correspond to application of  $\hat{\mathbf{B}}$  to *both* the primary and secondary functors, followed by function application:

$$(41) \quad \frac{\frac{\Delta}{x/(y/z)} \quad \frac{\Gamma}{y/w}}{\frac{(x/(w/z))/((y/z)/(w/z))}{x/(w/z)} \hat{\mathbf{B}}} \hat{\mathbf{B}}$$

As with  $\mathbf{B}^n$ ,  $\mathbf{D}^{n \geq 1}$  can be derived by iterative application of  $\hat{\mathbf{B}}$  to both primary and secondary functors.

Because  $\mathbf{B}$  can be derived from  $\hat{\mathbf{B}}$ , clause (iii) of (35) is equivalent to the following:

$$(42) \quad \text{If } \beta \text{ is not in Eisner-NF, then} \\ NF(\beta) = \langle FA, \langle \hat{\mathbf{B}}, \beta_1 \rangle, \beta_2 \rangle, \text{ such that} \\ NF(\alpha) = \langle S, \beta_1, NF(\langle T, \beta_2, \gamma \rangle) \rangle$$

Interpreted in terms of  $\hat{\mathbf{B}}$ , both  $\mathbf{B}$  and  $\mathbf{D}$  involve application of  $\hat{\mathbf{B}}$  to the primary functor. It follows that Theorem I applies directly to  $\mathbf{D}$  simply by virtue of the equivalence between binary  $\mathbf{B}$  and unary- $\hat{\mathbf{B}}$ +FA.

Eisner's NF constraints can then be reinterpreted as a constraint on  $\hat{\mathbf{B}}$  requiring its output to be an inert result category. We represent this in terms of the  $\hat{\mathbf{B}}$ -rules introducing an inert slash, indicated with “!” (adopting the convention from OpenCCG):

$$(43) \quad x/y : f_{xy} \Rightarrow (x/!z)/(y/!z) : \lambda h_{zy} \lambda x_z f h x$$

Hence, both binary  $\mathbf{B}$  and  $\mathbf{D}$  return inert functors:

$$(44) \quad \frac{\frac{\Delta}{x/y} \quad \frac{\Gamma}{y/z}}{\frac{(x/!z)/(y/!z)}{x/!z} \hat{\mathbf{B}}} \hat{\mathbf{B}}$$

$$(45) \quad \frac{\frac{\Delta}{x/(y/z)} \quad \frac{\Gamma}{y/w}}{\frac{(x/!(w/z))/((y/z)/!(w/z))}{x/!(w/z)} \hat{\mathbf{B}}} \hat{\mathbf{B}}$$

The binary substitution ( $\mathbf{S}$ ) combinator can be similarly incorporated into the system. Unary substitution  $\hat{\mathbf{S}}$  is like  $\hat{\mathbf{B}}$  except that it introduces a slash on only the argument-side of the input functor. We stipulate that  $\hat{\mathbf{S}}$  returns a category with inert slashes:

$$(46) \quad (\hat{\mathbf{S}}) \quad (x/y)/z \Rightarrow (x/!z)/(y/!z)$$

$\mathbf{T}$  is by definition unary. It follows that all the binary rules in CCG (including the  $\mathbf{D}$ -rules) can be reduced to (iterated) instantiations of the unary combinators  $\hat{\mathbf{B}}$ ,  $\hat{\mathbf{S}}$ , or  $\mathbf{T}$  plus function application.

This provides a basis for CCG in which all combinatory rules are derived from unary  $\hat{\mathbf{B}}$ ,  $\hat{\mathbf{S}}$ , and  $\mathbf{T}$ .

### 4.3 A Logical Basis for Eisner Normal Form

The previous section shows that deriving CCG rules from unary combinators allows us to derive the  $\mathbf{D}$ -rules while preserving Eisner NF. In this section, we present an alternate formulation of Eisner NF with Baldridge's (2002) CTL basis for CCG. This formulation allows us to derive the  $\mathbf{D}$ -rules as before, and does so in a way that seamlessly integrates with Baldridge's system of modalized functors.

In CTL,  $\mathbf{B}_\diamond$  and  $\mathbf{B}_\times$  are proofs derived via structural rules that allow associativity and permutation of symbols within a sequent, in combination with the slash introduction and elimination rules of the base logic. To control application of these rules, Baldridge keys them to binary modal operators  $\diamond$  (for associativity) and  $\times$  (for permutation). Given these,  $\mathbf{B}$  is proven in (47):

$$(47) \quad \frac{\frac{\Delta \vdash x/\diamond y \quad \Gamma \vdash y/\diamond z \quad [a \vdash z]}{(\Gamma \circ_\diamond a_i) \vdash y} [\diamond E]}{(\Delta \circ_\diamond (\Gamma \circ_\diamond a_i)) \vdash x} [\diamond E]}{((\Delta \circ_\diamond \Gamma) \circ_\diamond a_i) \vdash x} [RA]}{(\Delta \circ_\diamond \Gamma) \vdash x/\diamond z} [\diamond I]$$

In a CCG ruleset compiled from such logics, a category must have an appropriately decorated slash in order to be the input to a rule. This means that rules apply universally, without language-specific

restrictions. Instead, restrictions can only be declared via modalities marked on lexical categories.

Unary  $\hat{\mathbf{B}}$  and the  $\mathbf{D}$  rules in 4.2 can be derived using the same logic. For example,  $>\hat{\mathbf{B}}$  can be derived as in (48):

$$(48) \quad \frac{\frac{\frac{\frac{\frac{\Delta \vdash x/y \quad [f \vdash y/z]^1 \quad [a \vdash z]^2}{(f_1 \circ_\circ a_2) \vdash y} [/E]}{(\Delta \circ_\circ (f_1 \circ_\circ a_2)) \vdash x} [/E]}{((\Delta \circ_\circ f_1) \circ_\circ a_2) \vdash x} [RA]}{(\Delta \circ_\circ f_1) \vdash x/z} [/I]}{\Delta \vdash (x/z)/\circ(y/z)} [/I]$$

The  $\mathbf{D}$  rules are also theorems of this system. For example, the proof for  $>\mathbf{D}$  applies (48) as a lemma to each of the primary and secondary functors:

$$(49) \quad \frac{\frac{\Delta \vdash x/\circ(y/z) \quad \Gamma \vdash y/w}{\Delta \vdash (x/\circ(w/z))/\circ((y/z)/\circ(w/z))} >\hat{\mathbf{B}} \quad \frac{\Gamma \vdash y/w}{\Gamma \vdash (y/z)/\circ(w/z)} >\hat{\mathbf{B}}}{(\Delta \circ_\circ \Gamma) \vdash x/\circ(w/z)} [\hat{\mathbf{D}}]$$

$>\mathbf{D}_{\circ_\times}$  involves an associative version of  $\hat{\mathbf{B}}$  applied to the primary functor (50), and a permutative version to the secondary functor (51).

$$(50) \quad \frac{\frac{\frac{\frac{\Delta \vdash x/\circ(y \setminus_\times z) \quad [f \vdash (y \setminus_\times z)]^1 \quad [g \vdash w \setminus_\times z]^2}{(f_1 \circ_\circ g_2) \vdash y \setminus_\times z} [/E]}{(\Delta \circ_\circ (f_1 \circ_\circ g_2)) \vdash x} [/E]}{((\Delta \circ_\circ f_1) \circ_\circ g_2) \vdash x} [RA]}{(\Delta \circ_\circ f_1) \vdash x/(w \setminus_\times z)} [/I]}{\Delta \vdash (x/(w \setminus_\times z))/\circ((y \setminus_\times z)/(w \setminus_\times z))} [/I]$$

$$(51) \quad \frac{\frac{\frac{\frac{\Gamma \vdash y/w \quad [a \vdash z]^1 \quad [f \vdash w \setminus_\times z]^2}{(a_1 \circ_\times f_2) \vdash w} [\setminus_\times E]}{(\Gamma \circ_\circ (a_1 \circ_\times f_2)) \vdash y} [LP]}{(a_1 \circ_\times (\Gamma \circ_\circ f_2)) \vdash y} [\setminus_\times I]}{(\Gamma \circ_\circ f_2) \vdash y \setminus_\times z} [/I]}{\Gamma \vdash (y \setminus_\times z)/(w \setminus_\times z)} [/I]$$

Rules for  $\mathbf{D}$  with appropriate modalities can therefore be incorporated seamlessly into CCG.

In the preceding subsection, we encoded Eisner NF with inert slashes. In Baldridge’s CTL basis for CCG, inert slashes are represented as functors seeking non-lexical arguments, represented as categories marked with an *antecedent-governed* feature,

reflecting the intuition that non-lexical arguments have to be “bound” by a superordinate functor.

This is based on an interpretation of antecedent-government as a unary modality  $\diamond_{ant}$  that allows structures marked by it to permute to the left or right periphery of a structure:<sup>6</sup>

$$(52) \quad \frac{((\Delta_a \circ_\times \diamond_{ant} \Delta_b) \circ_\times \Delta_c) \vdash x}{((\Delta_a \circ_\times \Delta_c) \circ_\times \diamond_{ant} \Delta_b) \vdash x} \quad [ARP]}{(\Delta_a \circ_\times (\diamond_{ant} \Delta_b \circ_\times \Delta_c)) \vdash x} \quad [ALP]}{(\diamond_{ant} \Delta_b \circ_\times (\Delta_a \circ_\times \Delta_c)) \vdash x}$$

Unlike permutation rules without  $\diamond_{ant}$ , these permutation rules can only be used in a proof when preceded by a hypothetical category marked with the  $\square_{ant}^\downarrow$  modality. The elimination rule for  $\square_{ant}^\downarrow$ -modalities introduces a corresponding  $\diamond$ -marked object in the resulting structure, feeding the rule:

$$(53) \quad \frac{\frac{\frac{\frac{[a \vdash \square_{ant}^\downarrow z]^1}{\diamond_{ant} a_1 \vdash z} [\square_{ant}^\downarrow E]}{\Delta \vdash x/y \quad (\diamond_{ant} a_1 \circ_\times \Gamma) \vdash y} [\setminus_\times E]}{(\Delta \circ_\times (\diamond_{ant} a_1 \circ_\times \Gamma)) \vdash x} [ALP]}{(\diamond_{ant} a_1 \circ_\times (\Delta \circ_\times \Gamma)) \vdash x} [ALP]}{(\Delta \circ_\times \Gamma) \vdash x \setminus_\times \diamond_{ant} \square_{ant}^\downarrow z} [\diamond E]}{(\Delta \circ_\times \Gamma) \vdash x \setminus_\times \diamond_{ant} \square_{ant}^\downarrow z} [\setminus_\times I]^2$$

Re-introduction of the  $[a \vdash \diamond_{ant} \square_{ant}^\downarrow z]^k$  hypothesis results in a functor the argument of which is marked with  $\diamond_{ant} \square_{ant}^\downarrow$ . Because lexical categories are not marked as such, the functor cannot take a lexical argument, and so is effectively an inert functor.

In Baldridge’s (2002) system, only proofs involving the ARP and ALP rules produce inert categories. In Eisner NF, all instances of  $\mathbf{B}$ -rules result in inert categories. This can be reproduced in Baldridge’s system simply by keying *all* structural rules to the *ant*-modality, the result being that all proofs involving structural rules result in inert functors.

As desired, the  $\mathbf{D}$ -rules result in inert categories as well. For example,  $>\mathbf{D}$  is derived as follows ( $\square_{ant}^\downarrow$  and  $\diamond_{ant}$  are abbreviated as  $\square^\downarrow$  and  $\diamond$ ):

<sup>6</sup>Note that the diamond operator used here is a syntactic operator, rather than a semantic operator as used in (16) above. The unary modalities used in CTL describe accessibility relationships between subtypes and supertypes of particular categories: in effect, they define feature hierarchies. See Moortgat (1997) and Oehrle (To Appear) for further explanation.



$$(54) \quad \frac{\frac{\Gamma \vdash y/\diamond w \quad \frac{[a \vdash \square^\downarrow(w/\diamond z)]^1}{\diamond a \vdash w/\diamond z} [\square^\downarrow E] \quad \frac{[b \vdash \square^\downarrow z]^2}{\diamond b \vdash z} [\square^\downarrow E]}{\diamond a \circ_\diamond \diamond b \vdash w} [\circ E]}{\frac{\Gamma \circ_\diamond (\diamond a \circ_\diamond \diamond b) \vdash y} [RA]}{((\Gamma \circ_\diamond \diamond a) \circ_\diamond \diamond b) \vdash y} [\circ E]} \frac{[c \vdash \diamond \square^\downarrow z]^3}{((\Gamma \circ_\diamond \diamond a) \circ_\diamond c) \vdash y} [\circ E]^2} {(\Gamma \circ_\diamond \diamond a) \vdash y/\diamond \diamond \square^\downarrow z} [\circ I]^3$$

$$(55) \quad \frac{\frac{\Delta \vdash x/\diamond (y/\diamond \diamond \square^\downarrow z) \quad \frac{\Gamma \circ_\diamond \diamond a \vdash y/\diamond \diamond \square^\downarrow z}{(\Delta \circ_\diamond (\Gamma \circ_\diamond \diamond a)) \vdash x} [RA]}{((\Delta \circ_\diamond \Gamma) \circ_\diamond \diamond a) \vdash x} [\circ E]^1}{((\Delta \circ_\diamond \Gamma) \circ_\diamond d) \vdash x} [\circ E]^1} {(\Delta \circ_\diamond \Gamma) \vdash x/\diamond \diamond \square^\downarrow (w/\diamond z)} [\circ I]^4$$

(54)-(55) can be used as a lemma corresponding to the CCG rule in (57):

$$(56) \quad \frac{\Delta \vdash x/\diamond (y/\diamond \diamond \square^\downarrow z) \quad \Gamma \vdash y/\diamond w}{(\Delta \circ_\diamond \Gamma) \vdash x/\diamond \diamond \square^\downarrow (w/\diamond z)} [D]$$

$$(57) \quad x/\diamond (y/\diamond^\downarrow z) \quad y/\diamond w \Rightarrow x/\diamond^\downarrow (w/\diamond z)$$

This means that all CCG rules compiled from the logic—which requires  $\diamond_{ant}$  to licence the structural rules necessary to prove the rules—return inert functors. Eisner NF thus falls out of the logic because all instances of **B**, **D**, and **S** produce inert categories. This in turns allows us to view Eisner NF as part of a theory of grammatical competence, in addition to being a useful technique for constraining parsing.

## 5 Conclusion

Including the **D**-combinator rules in the CCG rule set lets us capture several linguistic generalizations that lack satisfactory analyses in standard CCG. Furthermore, CCG augmented with **D** is compatible with Eisner NF (Eisner, 1996), a standard technique for controlling derivational ambiguity in CCG-parsers, and also with the modalized version of CCG (Baldrige and Kruijff, 2003). A consequence is that both the **D** rules and the NF constraints can be derived from a grammar-internal perspective. This extends CCG’s linguistic applicability without sacrificing efficiency.

Wittenburg (1987) originally proposed using rules based on **D** as a way to reduce spurious ambiguity, which he achieved by eliminating **B** rules entirely and replacing them with variations on **D**. Wittenburg notes that doing so produces as many instances of **D** as there are rules in the standard rule set. Our proposal retains **B** and **S**, but, thanks to Eisner NF, eliminates spurious ambiguity, a result that Wittenburg was not able to realize at the time.

Our approach can be incorporated into Eisner NF straightforwardly. However, Eisner NF disprefers incremental analyses by forcing right-corner analyses of long-distance dependencies, such as in (58):

$$(58) \quad (\text{What (does (Grommet (think (Tottie (said (Victor (knows (Wallace ate))))))))))?)$$

For applications that call for increased incrementality (e.g., aligning visual and spoken input incrementally (Kruijff et al., 2007)), CCG rules that do not produce inert categories can be derived a CTL basis that does not require  $\diamond_{ant}$  for associativity and permutation. The **D**-rules derived from this kind of CTL specification would allow for left-corner analyses of such dependencies with the competence grammar. An extracted element can “wrap around” the words intervening between it and its extraction site. For example, **D** would allow the following bracketing for the same example (while producing the same logical form):

$$(59) \quad ((((((((((\text{What does}) \text{Grommet}) \text{think}) \text{Tottie}) \text{said}) \text{Victor}) \text{knows}) \text{Wallace}) \text{ate})))))?)$$

Finally, the unary combinator basis for CCG provides an interesting additional specification for generating CCG rules. Like the CTL basis, the unary combinator basis can produce a much wider range of possible rules, such as **D** rules, that may be relevant for linguistic applications. Whichever basis is used, inclusion of the **D**-rules increases empirical coverage, while at the same time preserving CCG’s computational attractiveness.

## Acknowledgments

Thanks Mark Steedman for extensive comments and suggestions, and particularly for noting the relationship between the **D**-rules and unary  $\hat{\mathbf{B}}$ . Thanks also to Emmon Bach, Cem Bozsahin, Jason Eisner, Geert-Jan Kruijff and the ACL reviewers.

## References

- Farrell Ackerman and John Moore. 1999. Syntagmatic and Paradigmatic Dimensions of Causee Encodings. *Linguistics and Philosophy*, 24:1–44.
- Avery D. Andrews and Christopher D. Manning. 1999. *Complex Predicates and Information Spreading in LFG*. CSLI Publications, Palo Alto, California.
- Jason Baldrige and Geert-Jan Kruijff. 2003. Multi-Modal Combinatory Categorical Grammar. In *Proceedings of EACL 10*, pages 211–218.
- Jason Baldrige, Sudipta Chatterjee, Alexis Palmer, and Ben Wing. 2007. DotCCG and VisCCG: Wiki and Programming Paradigms for Improved Grammar Engineering with OpenCCG. In *Proceedings of GEAF 2007*.
- Jason Baldrige. 2002. *Lexically Specified Derivational Control in Combinatory Categorical Grammar*. Ph.D. thesis, University of Edinburgh.
- John Beavers. 2004. Type-inheritance Combinatory Categorical Grammar. In *Proceedings of COLING-04*, Geneva, Switzerland.
- Robert Borsley and Kersti Börjars, editors. To Appear. *Non-Transformational Syntax: A Guide to Current Models*. Blackwell.
- Cem Bozsahin. 1998. Deriving the Predicate-Argument Structure for a Free Word Order Language. In *Proceedings of COLING-ACL '98*.
- Stephen Clark and James Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4).
- Haskell B. Curry and Robert Feys. 1958. *Combinatory Logic*, volume 1. North Holland, Amsterdam.
- Jason Eisner. 1996. Efficient Normal-Form Parsing for Combinatory Categorical Grammars. In *Proceedings of the ACL 34*.
- Michael D Finnemann. 1982. *Aspects of the Spanish Causative Construction*. Ph.D. thesis, University of Minnesota.
- L. T. F. Gamut. 1991. *Logic, Language, and Meaning*, volume II. Chicago University Press.
- Jeroen Groenendijk and Martin Stokhof. 1997. Questions. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 19, pages 1055–1124. Elsevier Science, Amsterdam.
- Mark Hepple and Glyn Morrill. 1989. Parsing and Derivational Equivalence. In *Proceedings of EACL 4*.
- Julia Hockenmaier and Mark Steedman. 2002. Generative Models for Statistical Parsing with Combinatory Categorical Grammar. In *Proceedings of ACL 40*, pages 335–342, Philadelphia, PA.
- Pauline Jacobson. 1990. Raising as Function Composition. *Linguistics and Philosophy*, 13:423–475.
- Pauline Jacobson. 1999. Towards a Variable-Free Semantics. *Linguistics and Philosophy*, 22:117–184.
- Lauri Karttunen. 1989. Radical Lexicalism. In Mark Baltin and Anthony Kroch, editors, *Alternative Conceptions of Phrase Structure*. University of Chicago Press, Chicago.
- Angelika Kratzer. 1991. Modality. In Arnim von Stechow and Dieter Wunderlich, editors, *Semantics: An International Handbook of Contemporary Semantic Research*, pages 639–650. Walter de Gruyter, Berlin.
- Geert-Jan M. Kruijff, Pierre Lison, Trevor Benjamin, Henrik Jacobsson, and Nick Hawes. 2007. Incremental, Multi-Level Processing for Comprehending Situated Dialogue in Human-Robot Interaction. In *Language and Robots: Proceedings from the Symposium (LangRo'2007)*, Aveiro, Portugal.
- Joachim Lambek. 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–169.
- Marta Luján. 1980. Clitic Promotion and Mood in Spanish Verbal Complements. *Linguistics*, 18:381–484.
- Michael Moortgat. 1997. Categorical Type Logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 93–177. North Holland, Amsterdam.
- Richard T Oehrlé. To Appear. Multi-Modal Type Logical Grammar. In Boersley and Börjars (Boersley and Börjars, To Appear).
- Martin Pickering and Guy Barry. 1993. Dependency Categorical Grammar and Coordination. *Linguistics*, 31:855–902.
- David Reitter, Julia Hockenmaier, and Frank Keller. 2006. Priming Effects in Combinatory Categorical Grammar. In *Proceedings of EMNLP-2006*.
- Mark Steedman and Jason Baldrige. To Appear. Combinatory Categorical Grammar. In Boersley and Börjars (Boersley and Börjars, To Appear).
- Mark Steedman. 1996. *Surface Structure and Interpretation*. MIT Press.
- Mark Steedman. 2000. *The Syntactic Process*. MIT Press.
- Michael White and Jason Baldrige. 2003. Adapting Chart Realization to CCG. In *Proceedings of ENLG*.
- Michael White. 2006. Efficient Realization of Coordinate Structures in Combinatory Categorical Grammar. *Research on Language and Computation*, 4(1):39–75.
- Kent Wittenburg. 1987. Predictive Combinators: A Method for Efficient Processing of Combinatory Categorical Grammars. In *Proceedings of ACL 25*.
- Luke Zettlemoyer and Michael Collins. 2007. Online Learning of Relaxed CCG Grammars for Parsing to Logical Form. In *Proceedings of EMNLP-CoNLL 2007*.

# Parsing Noun Phrase Structure with CCG

David Vadas and James R. Curran

School of Information Technologies

University of Sydney

NSW 2006, Australia

{dvadas1, james}@it.usyd.edu.au

## Abstract

Statistical parsing of noun phrase (NP) structure has been hampered by a lack of gold-standard data. This is a significant problem for CCGbank, where binary branching NP derivations are often incorrect, a result of the automatic conversion from the Penn Treebank.

We correct these errors in CCGbank using a gold-standard corpus of NP structure, resulting in a much more accurate corpus. We also implement novel NER features that generalise the lexical information needed to parse NPs and provide important semantic information. Finally, evaluating against DepBank demonstrates the effectiveness of our modified corpus and novel features, with an increase in parser performance of 1.51%.

## 1 Introduction

Internal noun phrase (NP) structure is not recovered by a number of widely-used parsers, e.g. Collins (2003). This is because their training data, the Penn Treebank (Marcus et al., 1993), does not fully annotate NP structure. The flat structure described by the Penn Treebank can be seen in this example:

```
(NP (NN lung) (NN cancer) (NNS deaths))
```

CCGbank (Hockenmaier and Steedman, 2007) is the primary English corpus for Combinatory Categorical Grammar (CCG) (Steedman, 2000) and was created by a semi-automatic conversion from the Penn Treebank. However, CCG is a binary branching grammar, and as such, cannot leave NP structure underspecified. Instead, *all* NPs were made right-branching, as shown in this example:

```
(N  
  (N/N lung)  
  (N  
    (N/N cancer) (N deaths) ) )
```

This structure is correct for most English NPs and is the best solution that doesn't require manual re-annotation. However, the resulting derivations often contain errors. This can be seen in the previous example, where *lung cancer* should form a constituent, but does not.

The first contribution of this paper is to correct these CCGbank errors. We apply an automatic conversion process using the gold-standard NP data annotated by Vadas and Curran (2007a). Over a quarter of the sentences in CCGbank need to be altered, demonstrating the magnitude of the NP problem and how important it is that these errors are fixed.

We then run a number of parsing experiments using our new version of the CCGbank corpus. In particular, we implement new features using NER tags from the BBN Entity Type Corpus (Weischedel and Brunstein, 2005). These features are targeted at improving the recovery of NP structure, increasing parser performance by 0.64% F-score.

Finally, we evaluate against DepBank (King et al., 2003). This corpus annotates internal NP structure, and so is particularly relevant for the changes we have made to CCGbank. The CCG parser now recovers additional structure learnt from our NP corrected corpus, increasing performance by 0.92%. Applying the NER features results in a total increase of 1.51%.

This work allows parsers trained on CCGbank to model NP structure accurately, and then pass this crucial information on to downstream systems.

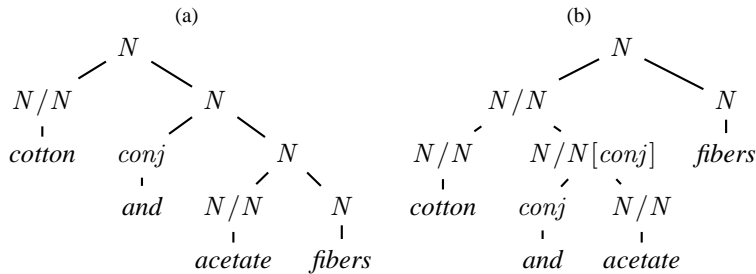


Figure 1: (a) Incorrect CCG derivation from Hockenmaier and Steedman (2007) (b) The correct derivation

## 2 Background

Parsing of NPs is typically framed as NP bracketing, where the task is limited to discriminating between left and right-branching NPs of three nouns only:

- *(crude oil) prices* – left-branching
- *world (oil prices)* – right-branching

Lauer (1995) presents two models to solve this problem: the adjacency model, which compares the association strength between words 1–2 to words 2–3; and the dependency model, which compares words 1–2 to words 1–3. Lauer (1995) experiments with a data set of 244 NPs, and finds that the dependency model is superior, achieving 80.7% accuracy.

Most NP bracketing research has used Lauer’s data set. Because it is a very small corpus, most approaches have been unsupervised, measuring association strength with counts from a separate large corpus. Nakov and Hearst (2005) use search engine hit counts and extend the query set with typographical markers. This results in 89.3% accuracy.

Recently, Vadas and Curran (2007a) annotated internal NP structure for the entire Penn Treebank, providing a large gold-standard corpus for NP bracketing. Vadas and Curran (2007b) carry out supervised experiments using this data set of 36,584 NPs, outperforming the Collins (2003) parser.

The Vadas and Curran (2007a) annotation scheme inserts NML and JJP brackets to describe the correct NP structure, as shown below:

```
(NP (NML (NN lung) (NN cancer) )
  (NNS deaths) )
```

We use these brackets to determine new gold-standard CCG derivations in Section 3.

### 2.1 Combinatory Categorical Grammar

Combinatory Categorical Grammar (CCG) (Steedman, 2000) is a type-driven, lexicalised theory of

grammar. Lexical categories (also called *supertags*) are made up of basic atoms such as *S* (Sentence) and *NP* (Noun Phrase), which can be combined to form complex categories. For example, a transitive verb such as *bought* (as in *IBM bought the company*) would have the category:  $(S \setminus NP) / NP$ . The slashes indicate the directionality of arguments, here two arguments are expected: an NP subject on the left; and an NP object on the right. Once these arguments are filled, a sentence is produced.

Categories are combined using combinatory rules such as forward and backward application:

$$X / Y \ Y \Rightarrow X \ (>) \quad (1)$$

$$Y \ X \setminus Y \Rightarrow X \ (<) \quad (2)$$

Other rules such as composition and type-raising are used to analyse some linguistic constructions, while retaining the canonical categories for each word. This is an advantage of CCG, allowing it to recover long-range dependencies without the need for post-processing, as is the case for many other parsers.

In Section 1, we described the incorrect NP structures in CCGbank, but a further problem that highlights the need to improve NP derivations is shown in Figure 1. When a conjunction occurs in an NP, a non-CCG rule is required in order to reach a parse:

$$conj \ N \Rightarrow N \quad (3)$$

This rule treats the conjunction in the same manner as a modifier, and results in the incorrect derivation shown in Figure 1(a). Our work creates the correct CCG derivation, shown in Figure 1(b), and removes the need for the grammar rule in (3).

Honnibal and Curran (2007) have also made changes to CCGbank, aimed at better differentiating between complements and adjuncts. PropBank (Palmer et al., 2005) is used as a gold-standard to inform these decisions, similar to the way that we use the Vadas and Curran (2007a) data.

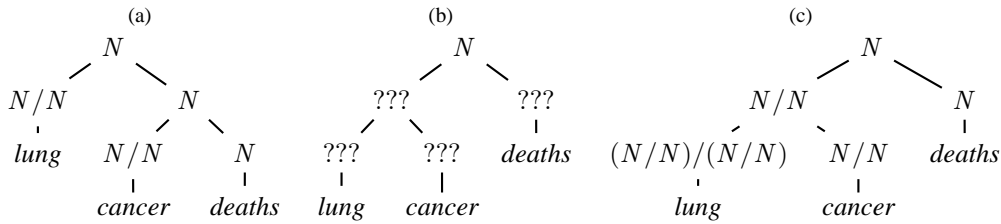


Figure 2: (a) Original right-branching CCGbank (b) Left-branching (c) Left-branching with new supertags

## 2.2 CCG parsing

The C&C CCG parser (Clark and Curran, 2007b) is used to perform our experiments, and to evaluate the effect of the changes to CCGbank. The parser uses a two-stage system, first employing a supertagger (Bangalore and Joshi, 1999) to propose lexical categories for each word, and then applying the CKY chart parsing algorithm. A log-linear model is used to identify the most probable derivation, which makes it possible to add the novel features we describe in Section 4, unlike a PCFG.

The C&C parser is evaluated on predicate-argument dependencies derived from CCGbank. These dependencies are represented as 5-tuples:  $\langle h_f, f, s, h_a, l \rangle$ , where  $h_f$  is the head of the predicate;  $f$  is the supertag of  $h_f$ ;  $s$  describes which argument of  $f$  is being filled;  $h_a$  is the head of the argument; and  $l$  encodes whether the dependency is local or long-range. For example, the dependency encoding *company* as the object of *bought* (as in *IBM bought the company*) is represented by:

$$\langle \text{bought}, (S \setminus NP_1) / NP_2, 2, \text{company}, - \rangle \quad (4)$$

This is a local dependency, where *company* is filling the second argument slot, the object.

## 3 Conversion Process

This section describes the process of converting the Vadas and Curran (2007a) data to CCG derivations. The tokens dominated by NML and JJP brackets in the source data are formed into constituents in the corresponding CCGbank sentence. We generate the two forms of output that CCGbank contains: AUTO files, which represent the tree structure of each sentence; and PARG files, which list the word–word dependencies (Hockenmaier and Steedman, 2005).

We apply one preprocessing step on the Penn Treebank data, where if multiple tokens are enclosed by brackets, then a NML node is placed around those

tokens. For example, we would insert the NML bracket shown below:

```
(NP (DT a) (-LRB- -LRB-)
  (NML (RB very) (JJ negative) )
  (-RRB- -RRB-) (NN reaction) )
```

This simple heuristic captures NP structure not explicitly annotated by Vadas and Curran (2007a).

The conversion algorithm applies the following steps for each NML or JJP bracket:

1. Identify the CCGbank *lowest spanning node*, the lowest constituent that covers all of the words in the NML or JJP bracket;
2. flatten the lowest spanning node, to remove the right-branching structure;
3. insert new left-branching structure;
4. identify heads;
5. assign supertags;
6. generate new dependencies.

As an example, we will follow the conversion process for the NML bracket below:

```
(NP (NML (NN lung) (NN cancer) )
  (NNS deaths) )
```

The corresponding lowest spanning node, which incorrectly has *cancer deaths* as a constituent, is shown in Figure 2(a). To flatten the node, we recursively remove brackets that partially overlap the NML bracket. Nodes that don't overlap at all are left intact. This process results in a list of nodes (which may or may not be leaves), which in our example is [*lung, cancer, deaths*]. We then insert the correct left-branching structure, shown in Figure 2(b). At this stage, the supertags are still incomplete.

Heads are then assigned using heuristics adapted from Hockenmaier and Steedman (2007). Since we are applying these to CCGbank NP structures rather than the Penn Treebank, the POS tag based heuristics are sufficient to determine heads accurately.

Finally, we assign supertags to the new structure. We want to make the minimal number of changes to the entire sentence derivation, and so the supertag of the dominating node is fixed. Categories are then propagated recursively down the tree. For a node with category  $X$ , its head child is also given the category  $X$ . The non-head child is always treated as an adjunct, and given the category  $X/X$  or  $X\backslash X$  as appropriate. Figure 2(c) shows the final result of this step for our example.

### 3.1 Dependency generation

The changes described so far have generated the new tree structure, but the last step is to generate new dependencies. We recursively traverse the tree, at each level creating a dependency between the heads of the left and right children. These dependencies are never long-range, and therefore easy to deal with. We may also need to change dependencies reaching from inside to outside the NP, if the head(s) of the NP have changed. In these cases we simply replace the old head(s) with the new one(s) in the relevant dependencies. The number of heads may change because we now analyse conjunctions correctly.

In our example, the original dependencies were:

$$\langle lung, N/N_1, 1, deaths, - \rangle \quad (5)$$

$$\langle cancer, N/N_1, 1, deaths, - \rangle \quad (6)$$

while after the conversion process, (5) becomes:

$$\langle lung, (N/N_1)/(N/N)_2, 2, cancer, - \rangle \quad (7)$$

To determine that the conversion process worked correctly, we manually inspected its output for unique tree structures in Sections 00–07. This identified problem cases to correct, such as those described in the following section.

### 3.2 Exceptional cases

Firstly, when the lowest spanning node covers the NML or JJP bracket exactly, no changes need to be made to CCGbank. These cases occur when CCGbank already received the correct structure during the original conversion process. For example, brackets separating a possessive from its possessor were detected automatically.

A more complex case is conjunctions, which do not follow the simple head/adjunct method of assigning supertags. Instead, conjuncts are identified

during the head-finding stage, and then assigned the supertag dominating the entire coordination. Intervening non-conjunct nodes are given the same category with the *conj* feature, resulting in a derivation that can be parsed with the standard CCGbank binary coordination rules:

$$conj\ X \Rightarrow X[conj] \quad (8)$$

$$X\ X[conj] \Rightarrow X \quad (9)$$

The derivation in Figure 1(b) is produced by these corrections to coordination derivations. As a result, applications of the non-CCG rule shown in (3) have been reduced from 1378 to 145 cases.

Some POS tags require special behaviour. Determiners and possessive pronouns are both usually given the supertag  $NP[nb]/N$ , and this should not be changed by the conversion process. Accordingly, we do not alter tokens with POS tags of DT and PRP\$. Instead, their sibling node is given the category  $N$  and their parent node is made the head. The parent's sibling is then assigned the appropriate adjunct category (usually  $NP\backslash NP$ ). Tokens with punctuation POS tags<sup>1</sup> do not have their supertag changed either.

Finally, there are cases where the lowest spanning node covers a constituent that should not be changed. For example, in the following NP:

```
(NP
  (NML (NN lower) (NN court) )
  (JJ final) (NN ruling) )
```

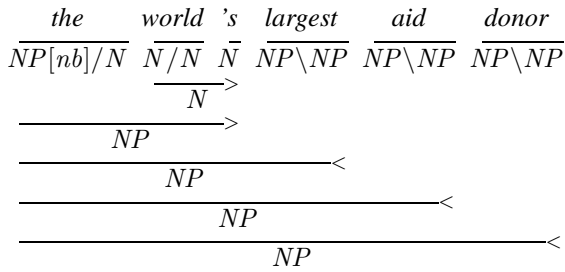
with the original CCGbank lowest spanning node:

```
(N (N/N lower)
  (N (N/N court)
    (N (N/N final) (N ruling) ) ) )
```

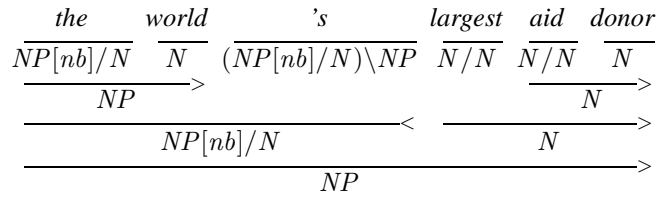
the final ruling node should not be altered.

It may seem trivial to process in this case, but consider a similarly structured NP: lower court ruling that the U.S. can bar the use of... Our minimalist approach avoids reanalysing the many linguistic constructions that can be dominated by NPs, as this would reinvent the creation of CCGbank. As a result, we only flatten those constituents that partially overlap the NML or JJP bracket. The existing structure and dependencies of other constituents are retained. Note that we are still converting every NML and JJP bracket, as even in the subordinate clause example, only the structure around lower court needs to be altered.

<sup>1</sup>period, comma, colon, and left and right bracket.



(a)



(b)

Figure 3: CCGbank derivations for possessives

	#	%
Possessive	224	43.75
Left child contains DT/PRP\$	87	16.99
Couldn't assign to non-leaf	66	12.89
Conjunction	35	6.84
Automatic conversion was correct	26	5.08
Entity with internal brackets	23	4.49
DT	22	4.30
NML/JJP bracket is an error	12	2.34
Other	17	3.32
Total	512	100.00

Table 1: Manual analysis

### 3.3 Manual annotation

A handful of problems that occurred during the conversion process were corrected manually. The first indicator of a problem was the presence of a possessive. This is unexpected, because possessives were already bracketed properly when CCGbank was originally created (Hockenmaier, 2003, §3.6.4). Secondly, a non-flattened node should not be assigned a supertag that it did not already have. This is because, as described previously, a non-leaf node could dominate any kind of structure. Finally, we expect the lowest spanning node to cover only the NML or JJP bracket and one more constituent to the right. If it doesn't, because of unusual punctuation or an incorrect bracket, then it may be an error. In all these cases, which occur throughout the corpus, we manually analysed the derivation and fixed any errors that were observed.

512 cases were flagged by this approach, or 1.90% of the 26,993 brackets converted to CCG. Table 1 shows the causes of these problems. The most common cause of errors was possessives, as the con-

version process highlighted a number of instances where the original CCGbank analysis was incorrect. An example of this error can be seen in Figure 3(a), where the possessive doesn't take any arguments. Instead, *largest aid donor* incorrectly modifies the NP one word at a time. The correct derivation after manual analysis is in (b).

The second-most common cause occurs when there is apposition inside the NP. This can be seen in Figure 4. As there is no punctuation on which to coordinate (which is how CCGbank treats most appositions) the best derivation we can obtain is to have *Victor Borge* modify the preceding NP.

The final step in the conversion process was to validate the corpus against the CCG grammar, first by those productions used in the existing CCGbank, and then against those actually licensed by CCG (with pre-existing ungrammaticalities removed). Sixteen errors were identified by this process and subsequently corrected by manual analysis.

In total, we have altered 12,475 CCGbank sentences (25.5%) and 20,409 dependencies (1.95%).

## 4 NER features

Named entity recognition (NER) provides information that is particularly relevant for NP parsing, simply because entities are nouns. For example, knowing that *Air Force* is an entity tells us that *Air Force contract* is a left-branching NP.

Vadas and Curran (2007a) describe using NE tags during the annotation process, suggesting that NER-based features will be helpful in a statistical model. There has also been recent work combining NER and parsing in the biomedical field. Lewin (2007) experiments with detecting base-NPs using NER information, while Buyko et al. (2007) use a CRF to identify

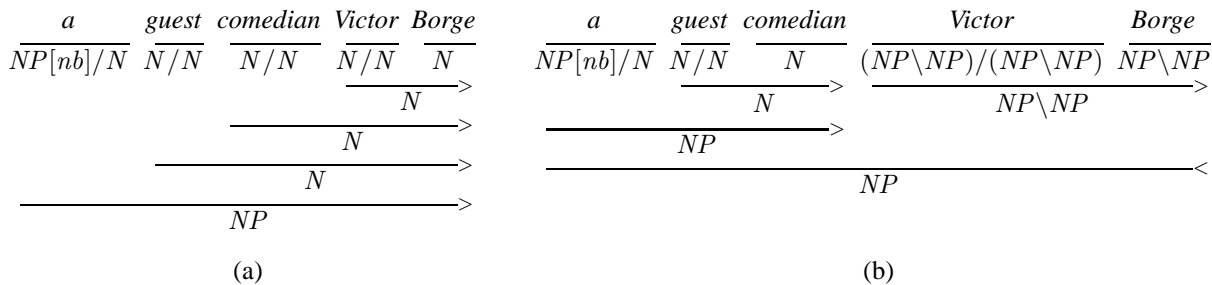


Figure 4: CCGbank derivations for apposition with DT

coordinate structure in biological named entities.

We draw NE tags from the BBN Entity Type Corpus (Weischedel and Brunstein, 2005), which describes 28 different entity types. These include the standard person, location and organization classes, as well person descriptions (generally occupations), NORP (National, Other, Religious or Political groups), and works of art. Some classes also have finer-grained subtypes, although we use only the coarse tags in our experiments.

Clark and Curran (2007b) has a full description of the C&C parser’s pre-existing features, to which we have added a number of novel NER-based features. Many of these features generalise the head words and/or POS tags that are already part of the feature set. The results of applying these features are described in Sections 5.3 and 6.

The first feature is a simple lexical feature, describing the NE tag of each token in the sentence. This feature, and all others that we describe here, are not active when the NE tag(s) are **O**, as there is no NER information from tokens that are not entities.

The next group of features is based on the local tree (a parent and two child nodes) formed by every grammar rule application. We add a feature where the rule being applied is combined with the parent’s NE tag. For example, when joining two constituents<sup>2</sup>:  $\langle \text{five}, \text{CD}, \text{CARD}, N/N \rangle$  and  $\langle \text{Europeans}, \text{NNPS}, \text{NORP}, N \rangle$ , the feature is:

$$N \rightarrow N/N N + \text{NORP}$$

as the head of the constituent is *Europeans*.

In the same way, we implement features that combine the grammar rule with the child nodes. There are already features in the model describing each combination of the children’s head words and POS tags, which we extend to include combinations with

<sup>2</sup>These 4-tuples are the node’s head, POS, NE, and supertag.

the NE tags. Using the same example as above, one of the new features would be:

$$N \rightarrow N/N N + \text{CARD} + \text{NORP}$$

The last group of features is based on the NE category spanned by each constituent. We identify constituents that dominate tokens that all have the same NE tag, as these nodes will not cause a “crossing bracket” with the named entity. For example, the constituent *Force contract*, in the NP *Air Force contract*, spans two different NE tags, and should be penalised by the model. *Air Force*, on the other hand, only spans **ORG** tags, and should be preferred accordingly.

We also take into account whether the constituent spans the *entire* named entity. Combining these nodes with others of different NE tags should *not* be penalised by the model, as the NE must combine with the rest of the sentence at some point.

These NE spanning features are implemented as the grammar rule in combination with the parent node or the child nodes. For the former, one feature is active when the node spans the entire entity, and another is active in other cases. Similarly, there are four features for the child nodes, depending on whether neither, the left, the right or both nodes span the entire NE. As an example, if the *Air Force* constituent were being joined with *contract*, then the child feature would be:

$$N \rightarrow N/N N + \text{LEFT} + \text{ORG} + \text{O}$$

assuming that there are more **O** tags to the right.

## 5 Experiments

Our experiments are run with the C&C CCG parser (Clark and Curran, 2007b), and will evaluate the changes made to CCGbank, as well as the effectiveness of the NER features. We train on Sections 02-21, and test on Section 00.



	PREC	RECALL	F-SCORE
Original	91.85	92.67	92.26
NP corrected	91.22	92.08	91.65

Table 2: Supertagging results

	PREC	RECALL	F-SCORE
Original	85.34	84.55	84.94
NP corrected	85.08	84.17	84.63

Table 3: Parsing results with gold-standard POS tags

## 5.1 Supertagging

Before we begin full parsing experiments, we evaluate on the supertagger alone. The supertagger is an important stage of the CCG parsing process, its results will affect performance in later experiments.

Table 2 shows that F-score has dropped by 0.61%. This is not surprising, as the conversion process has increased the ambiguity of supertags in NPs. Previously, a bare NP could only have a sequence of  $N/N$  tags followed by a final  $N$ . There are now more complex possibilities, equal to the Catalan number of the length of the NP.

## 5.2 Initial parsing results

We now compare parser performance on our NP corrected version of the corpus to that on original CCGbank. We are using the normal-form parser model and report labelled precision, recall and F-score for all dependencies. The results are shown in Table 3.

The F-score drops by 0.31% in our new version of the corpus. However, this comparison is not entirely fair, as the original CCGbank test data does not include the NP structure that the NP corrected model is being evaluated on. Vadas and Curran (2007a) experienced a similar drop in performance on Penn Treebank data, and noted that the F-score for  $NML$  and  $JJP$  brackets was about 20% lower than the overall figure. We suspect that a similar effect is causing the drop in performance here.

Unfortunately, there are no explicit  $NML$  and  $JJP$  brackets to evaluate on in the CCG corpus, and so an NP structure only figure is difficult to compute. Recall can be calculated by marking those dependencies altered in the conversion process, and evaluating only on them. Precision cannot be measured in this

	PREC	RECALL	F-SCORE
Original	83.65	82.81	83.23
NP corrected	83.31	82.33	82.82

Table 4: Parsing results with automatic POS tags

	PREC	RECALL	F-SCORE
Original	86.00	85.15	85.58
NP corrected	85.71	84.83	85.27

Table 5: Parsing results with NER features

way, as NP dependencies remain undifferentiated in parser output. The result is a recall of 77.03%, which is noticeably lower than the overall figure.

We have also experimented with using automatically assigned POS tags. These tags are accurate with an F-score of 96.34%, with precision 96.20% and recall 96.49%. Table 4 shows that, unsurprisingly, performance is lower without the gold-standard data. The NP corrected model drops an additional 0.1% F-score over the original model, suggesting that POS tags are particularly important for recovering internal NP structure. Evaluating NP dependencies only, in the same manner as before, results in a recall figure of 75.21%.

## 5.3 NER features results

Table 5 shows the results of adding the NER features we described in Section 4. Performance has increased by 0.64% on both versions of the corpora. It is surprising that the NP corrected increase is not larger, as we would expect the features to be less effective on the original CCGbank. This is because incorrect right-branching NPs such as *Air Force contract* would introduce noise to the NER features.

Table 6 presents the results of using automatically assigned POS and NE tags, i.e. parsing raw text. The NER tagger achieves 84.45% F-score on all non-O classes, with precision being 78.35% and recall 91.57%. We can see that parsing F-score has dropped by about 2% compared to using gold-standard POS and NER data, however, the NER features still improve performance by about 0.3%.

	PREC	RECALL	F-SCORE
Original	83.92	83.06	83.49
NP corrected	83.62	82.65	83.14

Table 6: Parsing results with automatic POS and NE tags

## 6 DepBank evaluation

One problem with the evaluation in the previous section, is that the original CCGbank is not expected to recover internal NP structure, making its task easier and inflating its performance. To remove this variable, we carry out a second evaluation against the Briscoe and Carroll (2006) reannotation of DepBank (King et al., 2003), as described in Clark and Curran (2007a). Parser output is made similar to the grammatical relations (GRs) of the Briscoe and Carroll (2006) data, however, the conversion remains complex. Clark and Curran (2007a) report an upper bound on performance, using gold-standard CCGbank dependencies, of 84.76% F-score.

This evaluation is particularly relevant for NPs, as the Briscoe and Carroll (2006) corpus *has* been annotated for internal NP structure. With our new version of CCGbank, the parser will be able to recover these GRs correctly, where before this was unlikely.

Firstly, we show the figures achieved using gold-standard CCGbank derivations in Table 7. In the NP corrected version of the corpus, performance has increased by 1.02% F-score. This is a reversal of the results in Section 5, and demonstrates that correct NP structure improves parsing performance, rather than reduces it. Because of this increase to the upper bound of performance, we are now even closer to a true formalism-independent evaluation.

We now move to evaluating the C&C parser itself and the improvement gained by the NER features. Table 8 show our results, with the NP corrected version outperforming original CCGbank by 0.92%. Using the NER features has also caused an increase in F-score, giving a total improvement of 1.51%. These results demonstrate how successful the correcting of NPs in CCGbank has been.

Furthermore, the performance increase of 0.59% on the NP corrected corpus is more than the 0.25% increase on the original. This demonstrates that NER features are particularly helpful for NP structure.

	PREC	RECALL	F-SCORE
Original	86.86	81.61	84.15
NP corrected	87.97	82.54	85.17

Table 7: DepBank gold-standard evaluation

	PREC	RECALL	F-SCORE
Original	82.57	81.29	81.92
NP corrected	83.53	82.15	82.84
Original, NER	82.87	81.49	82.17
NP corrected, NER	84.12	82.75	83.43

Table 8: DepBank evaluation results

## 7 Conclusion

The first contribution of this paper is the application of the Vadas and Curran (2007a) data to Combinatory Categorical Grammar. Our experimental results have shown that this more accurate representation of CCGbank’s NP structure increases parser performance. Our second major contribution is the introduction of novel NER features, a source of semantic information previously unused in parsing.

As a result of this work, internal NP structure is now recoverable by the C&C parser, a result demonstrated by our total performance increase of 1.51% F-score. Even when parsing raw text, without gold standard POS and NER tags, our approach has resulted in performance gains.

In addition, we have made possible further increases to NP structure accuracy. New features can now be implemented and evaluated in a CCG parsing context. For example, bigram counts from a very large corpus have already been used in NP bracketing, and could easily be applied to parsing. Similarly, additional supertagging features can now be created to deal with the increased ambiguity in NPs.

Downstream NLP components can now exploit the crucial information in NP structure.

## Acknowledgements

We would like to thank Mark Steedman and Matthew Honnibal for help with converting the NP data to CCG; and the anonymous reviewers for their helpful feedback. This work has been supported by the Australian Research Council under Discovery Project DP0665973.

## References

- Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.
- Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 41–48. Sydney, Australia.
- Ekaterina Buyko, Katrin Tomanek, and Udo Hahn. 2007. Resolution of coordination ellipses in biological named entities with conditional random fields. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-2007)*, pages 163–171. Melbourne, Australia.
- Stephen Clark and James R. Curran. 2007a. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 248–255. Prague, Czech Republic.
- Stephen Clark and James R. Curran. 2007b. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.
- Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.
- Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorical Grammar*. Ph.D. thesis, University of Edinburgh.
- Julia Hockenmaier and Mark Steedman. 2005. CCGbank manual. Technical Report MS-CIS-05-09, Department of Computer and Information Science, University of Pennsylvania.
- Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.
- Matthew Honnibal and James R. Curran. 2007. Improving the complement/adjunct distinction in CCGbank. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-07)*, pages 210–217. Melbourne, Australia.
- Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC700 dependency bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*. Budapest, Hungary.
- 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*, pages 47–54. Cambridge, MA.
- Ian Lewin. 2007. BaseNPs that contain gene names: domain specificity and genericity. In *Biological, translational, and clinical language processing workshop*, pages 163–170. Prague, Czech Republic.
- 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. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL-05)*, pages 17–24. Ann Arbor, MI.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.
- Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.
- David Vadas and James R. Curran. 2007a. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07)*, pages 240–247. Prague, Czech Republic.
- David Vadas and James R. Curran. 2007b. Large-scale supervised models for noun phrase bracketing. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-2007)*, pages 104–112. Melbourne, Australia.
- Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report.

# Sentence Simplification for Semantic Role Labeling

David Vickrey and Daphne Koller

Stanford University

Stanford, CA 94305-9010

{dvickrey, koller}@cs.stanford.edu

## Abstract

Parse-tree paths are commonly used to incorporate information from syntactic parses into NLP systems. These systems typically treat the paths as atomic (or nearly atomic) features; these features are quite sparse due to the immense variety of syntactic expression. In this paper, we propose a general method for learning how to iteratively simplify a sentence, thus decomposing complicated syntax into small, easy-to-process pieces. Our method applies a series of hand-written transformation rules corresponding to basic syntactic patterns — for example, one rule “depassivizes” a sentence. The model is parameterized by *learned* weights specifying preferences for some rules over others. After applying all possible transformations to a sentence, we are left with a set of candidate simplified sentences. We apply our simplification system to semantic role labeling (SRL). As we do not have labeled examples of correct simplifications, we use labeled training data for the SRL task to jointly learn both the weights of the simplification model and of an SRL model, treating the simplification as a hidden variable. By extracting and labeling simplified sentences, this combined simplification/SRL system better generalizes across syntactic variation. It achieves a statistically significant 1.2% F1 measure increase over a strong baseline on the Conll-2005 SRL task, attaining near-state-of-the-art performance.

## 1 Introduction

In *semantic role labeling* (SRL), given a sentence containing a *target verb*, we want to label the semantic arguments, or roles, of that verb. For the verb “eat”, a correct labeling of “Tom ate a salad” is {ARG0(Eater)=“Tom”, ARG1(Food)=“salad”}.

Current semantic role labeling systems rely primarily on syntactic features in order to identify and

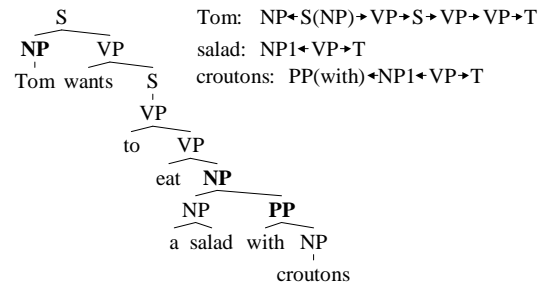


Figure 1: Parse with path features for verb “eat”.

classify roles. Features derived from a syntactic parse of the sentence have proven particularly useful (Gildea & Jurafsky, 2002). For example, the syntactic subject of “give” is nearly always the Giver. Path features allow systems to capture both general patterns, e.g., that the ARG0 of a sentence tends to be the subject of the sentence, and specific usage, e.g., that the ARG2 of “give” is often a post-verbal prepositional phrase headed by “to”. An example sentence with extracted path features is shown in Figure 1.

A major problem with this approach is that the path from an argument to the verb can be quite complicated. In the sentence “He expected to receive a prize for winning,” the path from “win” to its ARG0, “he”, involves the verbs “expect” and “receive” and the preposition “for.” The corresponding path through the parse tree likely occurs a relatively small number of times (or not at all) in the training corpus. If the test set contained exactly the same sentence but with “expected” replaced by “did not expect” we would extract a different parse path feature; therefore, as far as the classifier is concerned, the syntax of the two sentences is totally unrelated.

In this paper we learn a mapping from full, complicated sentences to simplified sentences. For example, given a correct parse, our system simplifies the above sentence with target verb “win” to “He won.” Our method combines hand-written syntactic simplification rules with machine learning, which

determines which rules to prefer. We then use the output of the simplification system as input to a SRL system that is trained to label simplified sentences.

Compared to previous SRL models, our model has several qualitative advantages. First, we believe that the simplification process, which represents the syntax as a set of local syntactic transformations, is more linguistically satisfying than using the entire parse path as an atomic feature. Improving the simplification process mainly involves adding more linguistic knowledge in the form of simplification rules. Second, labeling simple sentences is much easier than labeling raw sentences and allows us to generalize more effectively across sentences with differing syntax. This is particularly important for verbs with few labeled training instances; using training examples as efficiently as possible can lead to considerable gains in performance. Third, our model is very effective at sharing information across verbs, since most of our simplification rules apply equally well regardless of the target verb.

A major difficulty in learning to simplify sentences is that we do not have labeled data for this task. To address this problem, we simultaneously train our simplification system and the SRL system. We treat the correct simplification as a hidden variable, using labeled SRL data to guide us towards “more useful” simplifications. Specifically, we train our model discriminatively to predict the correct role labeling assignment given an input sentence, treating the simplification as a hidden variable.

Applying our combined simplification/SRL model to the Conll 2005 task, we show a significant improvement over a strong baseline model. Our model does best on verbs with little training data and on instances with paths that are rare or have never been seen before, matching our intuitions about the strengths of the model. Our model outperforms all but the best few Conll 2005 systems, each of which uses multiple different automatically-generated parses (which would likely improve our model).

## 2 Sentence Simplification

We will begin with an example before describing our model in detail. Figure 2 shows a series of transformations applied to the sentence “I was not given a chance to eat,” along with the interpretation of each transformation. Here, the target verb is “eat.”

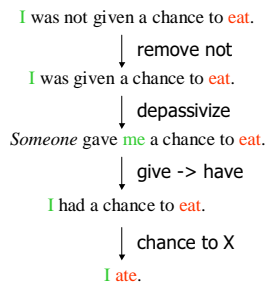


Figure 2: Example simplification

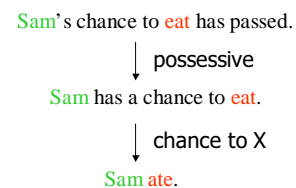


Figure 3: Shared simplification structure

There are several important things to note. First, many of the steps do lose some semantic information; clearly, having a chance to eat is not the same as eating. However, since we are interested only in labeling the core arguments of the verb (which in this case is simply the Eater, “I”), it is not important to maintain this information. Second, there is more than one way to choose a set of rules which lead to the desired final sentence “I ate.” For example, we could have chosen to include a rule which went directly from the second step to the fourth. In general, the rules were designed to allow as much reuse of rules as possible. Figure 3 shows the simplification of “Sam’s chance to eat has passed” (again with target verb “eat”); by simplifying both of these sentences as “X had a chance to Y”, we are able to use the same final rule in both cases.

Of course, there may be more than one way to simplify a sentence for a given rule set; this ambiguity is handled by learning which rules to prefer.

In this paper, we use simplification to mean something which is closer to *canonicalization* than *summarization*. Thus, given an input sentence, our goal is not to produce a single shortened sentence which contains as much information from the original sentence as possible. Rather, the goal is, for *each* verb in the sentence, to produce a “simple” sentence which is in a particular canonical form (described below) relative to that verb.

## 3 Transformation Rules

A *transformation rule* takes as input a parse tree and produces as output a different, changed parse tree. Since our goal is to produce a simplified version of the sentence, the rules are designed to bring all sentences toward the same common format.

A rule (see left side of Figure 4) consists of two

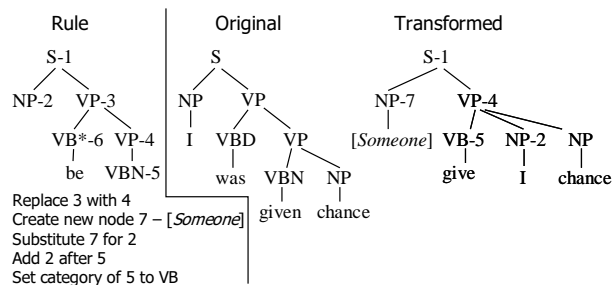


Figure 4: Rule for depassivizing a sentence

parts. The first is a “tree regular expression” which is most simply viewed as a tree fragment with optional constraints at each node. The rule assigns numbers to each node which are referred to in the second part of the rule. Formally, a rule node  $X$  matches a parse-tree node  $A$  if: (1) All constraints of node  $X$  (e.g., constituent category, head word, etc.) are satisfied by node  $A$ . (2) For each child node  $Y$  of  $X$ , there is a child  $B$  of  $A$  that matches  $Y$ ; two children of  $X$  cannot be matched to the same child  $B$ . There are no other requirements.  $A$  can have other children besides those matched, and leaves of the rule pattern can match to internal nodes of the parse (corresponding to entire phrases in the original sentence). For example, the same rule is used to simplify both “I had a chance to eat,” and “I had a chance to eat a sandwich,” (into “I ate,” and “I ate a sandwich,”). The insertion of the phrase “a sandwich” does not prevent the rule from matching.

The second part of the rule is a series of simple steps that are applied to the matched nodes. For example, one type of simple step applied to the pair of nodes  $(X,Y)$  removes  $X$  from its current parent and adds it as the final child of  $Y$ . Figure 4 shows the depassivizing rule and the result of applying it to the sentence “I was given a chance.” The transformation steps are applied sequentially from top to bottom. Note that any nodes not matched are unaffected by the transformation; they remain where they are relative to their parents. For example, “chance” is not matched by the rule, and thus remains as a child of the VP headed by “give.”

There are two significant pieces of “machinery” in our current rule set. The first is the idea of a *floating node*, used for locating an argument within a subordinate clause. For example, in the phrases “The cat that ate the mouse”, “The seed that the mouse ate”, and “The person we gave the gift to”, the modified nouns (“cat”, “seed”, and “person”, respectively) all

Rule Category	#	Original	Simplified
Sentence normalization	24	Thursday, I <u>slept</u> .	I <u>slept</u> Thursday.
Sentence extraction	4	I said he <u>slept</u> .	He <u>slept</u> .
Passive	5	I was <u>hit</u> by a car.	A car <u>hit</u> me.
Misc Collapsing/Rewriting	20	John, a lawyer, ...	John is a lawyer.
Conjunctions	8	I <u>ate</u> and slept.	I <u>ate</u> .
Verb Collapsing/Rewriting	14	I must <u>eat</u> .	I <u>eat</u> .
Verb Raising/Control (basic)	17	I <u>want</u> to eat.	I <u>eat</u> .
Verb RC (ADJP/ADVP)	6	I am likely to <u>eat</u> .	I <u>eat</u> .
Verb RC (Noun)	7	I have a chance to <u>eat</u> .	I <u>eat</u> .
Modified nouns	8	The food I <u>ate</u> ...	Float(The food) I <u>ate</u> .
Floating nodes	5	Float(The food) I <u>ate</u> .	I <u>ate</u> the food.
Inverted sentences	7	Nor will I <u>eat</u> .	I will <u>eat</u> .
Questions	7	Will I <u>eat</u> ?	I will <u>eat</u> .
Possessive	7	John’s chance to <u>eat</u> ...	John has a chance to <u>eat</u> .
Verb acting as PP/NP	7	<u>Including</u> tax, the total...	The total <u>includes</u> tax.
“Make” rewrites	8	Salt makes food tasty.	Food is tasty.

Table 1: Rule categories with sample simplifications. Target verbs are underlined.

should be placed in different positions in the subordinate clauses (subject, direct object, and object of “to”) to produce the phrases “The cat ate the mouse,” “The mouse ate the seed”, and “We gave the gift to the person.” We handle these phrases by placing a floating node in the subordinate clause which points to the argument; other rules try to place the floating node into each possible position in the sentence.

The second construct is a system for keeping track of whether a sentence has a subject, and if so, what it is. A subset of our rule set normalizes the input sentence by moving modifiers after the verb, leaving either a single phrase (the subject) or nothing before the verb. For example, the sentence “Before leaving, I ate a sandwich,” is rewritten as “I ate a sandwich before leaving.” In many cases, keeping track of the presence or absence of a subject greatly reduces the set of possible simplifications.

Altogether, we currently have 154 (mostly unlexicalized) rules. Our general approach was to write very conservative rules, i.e., avoid making rules with low precision, as these can quickly lead to a large blowup in the number of generated simple sentences. Table 1 shows a summary of our rule-set, grouped by type. Note that each row lists only one possible sentence and simplification rule from that

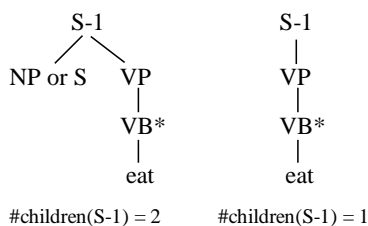


Figure 5: Simple sentence constraints for “eat”

category; many of the categories handle a variety of syntax patterns. The two examples without target verbs are helper transformations; in more complex sentences, they can enable further simplifications. Another thing to note is that we use the terms Raising/Control (RC) very loosely to mean situations where the subject of the target verb is displaced, appearing as the subject of another verb (see table).

Our rule set was developed by analyzing performance and coverage on the PropBank WSJ training set; neither the development set nor (of course) the test set were used during rule creation.

#### 4 Simple Sentence Production

We now describe how to take a set of rules and produce a set of candidate simple sentences. At a high level, the algorithm is very simple. We maintain a set of derived parses  $S$  which is initialized to contain only the original, untransformed parse. One iteration of the algorithm consists of applying every possible matching transformation rule to every parse in  $S$ , and adding all resulting parses to  $S$ . With carefully designed rules, repeated iterations are guaranteed to converge; that is, we eventually arrive at a set  $\hat{S}$  such that if we apply an iteration of rule application to  $\hat{S}$ , no new parses will be added. Note that we simplify the whole sentence without respect to a particular verb. Thus, this process only needs to be done once per sentence (not once per verb).

To label arguments of a particular target verb, we remove any parse from our set which does not match one of the two templates in Figure 5 (for verb “eat”). These select simple sentences that have all non-subject modifiers moved to the predicate and “eat” as the main verb. Note that the constraint VB\* indicates any terminal verb category (e.g., VBN, VBD, etc.) A parse that matches one of these templates is called a *valid simple sentence*; this is exactly the canonicalized version of the sentence which our simplification rules are designed to produce.

This procedure is quite expensive; we have to copy the entire parse tree at each step, and in general, this procedure could generate an exponential number of transformed parses. The first issue can be solved, and the second alleviated, using a dynamic-programming data structure similar to the one used to store parse forests (as in a chart parser). This data structure is not essential for exposition; we delay discussion until Section 7.

#### 5 Labeling Simple Sentences

For a particular sentence/target verb pair  $s, v$ , the output from the previous section is a set  $S^{sv} = \{t_i^{sv}\}_i$  of valid simple sentences. Although labeling a simple sentence is easier than labeling the original sentence, there are still many choices to be made. There is one key assumption that greatly reduces the search space: in a simple sentence, only the subject (if present) and direct modifiers of the target verb can be arguments of that verb.

On the training set, we now extract a set of *role patterns*  $G^v = \{g_j^v\}_j$  for each verb  $v$ . For example, a common role pattern for “give” is that of “I gave him a sandwich”. We represent this pattern as  $g_1^{give} = \{ARG0 = Subject NP, ARG1 = Postverb NP2, ARG2 = Postverb NP1\}$ . Note that this is one atomic pattern; thus, we are keeping track not just of occurrences of particular roles in particular places in the simple sentence, but also how those roles co-occur with other roles.

For a particular simple sentence  $t_i^{sv}$ , we apply all extracted role patterns  $g_j^v$  to  $t_i^{sv}$ , obtaining a set of possible role labelings. We call a simple sentence/role labeling pair a *simple labeling* and denote the set of candidate simple labelings  $C^{sv} = \{c_k^{sv}\}_k$ . Note that a given pair  $t_i^{sv}, g_j^v$  may generate more than one simple labeling, if there is more than one way to assign the elements of  $g_j^v$  to constituents in  $t_i^{sv}$ . Also, for a sentence  $s$  there may be several simple labelings that lead to the same role labeling. In particular, there may be several simple labelings which assign the correct labels to all constituents; we denote this set  $K^{sv} \subseteq C^{sv}$ .

#### 6 Probabilistic Model

We now define our probabilistic model. Given a (possibly large) set of candidate simple labelings  $C^{sv}$ , we need to select a correct one. We assign a score to each candidate based on its features:

Rule = Depassivize  
 Pattern = { ARG0 = Subj NP, ARG1 = PV NP2, ARG2 = PV NP1 }  
 Role = ARG0, Head Word = John  
 Role = ARG1, Head Word = sandwich  
 Role = ARG2, Head Word = I  
 Role = ARG0, Category = NP  
 Role = ARG1, Category = NP  
 Role = ARG2, Category = NP  
 Role = ARG0, Position = Subject NP  
 Role = ARG1, Position = Postverb NP2  
 Role = ARG2, Position = Postverb NP1

Figure 6: Features for “John gave me a sandwich.”

which rules were used to obtain the simple sentence, which role pattern was used, and features about the assignment of constituents to roles. A log-linear model then assigns probability to each simple labeling equal to the normalized exponential of the score.

The first type of feature is which rules were used to obtain the simple sentence. These features are indicator functions for each possible rule. Thus, we do not currently learn anything about interactions between different rules. The second type of feature is an indicator function of the role pattern used to generate the labeling. This allows us to learn that “give” has a preference for the labeling {*ARG0 = Subject NP, ARG1 = Postverb NP2, ARG2 = Postverb NP1*}. Our final features are analogous to those used in semantic role labeling, but greatly simplified due to our use of simple sentences: head word of the constituent; category (i.e., constituent label); and position in the simple sentence. Each of these features is combined with the role assignment, so that each feature indicates a preference for a particular role assignment (i.e., for “give”, head word “sandwich” tends to be ARG1). For each feature, we have a verb-specific and a verb-independent version, allowing sharing across verbs while still permitting different verbs to learn different preferences. The set of extracted features for the sentence “I was given a sandwich by John” with simplification “John gave me a sandwich” is shown in Figure 6. We omit verb-specific features to save space. Note that we “stem” all pronouns (including possessive pronouns).

For each candidate simple labeling  $c_k^{sv}$  we extract a vector of features  $\mathbf{f}_k^{sv}$  as described above. We now define the probability of a simple labeling  $c_k^{sv}$  with respect to a weight vector  $\mathbf{w}$   $P(c_k^{sv}) = \frac{e^{\mathbf{w}^T \mathbf{f}_k^{sv}}}{\sum_{k'} e^{\mathbf{w}^T \mathbf{f}_{k'}^{sv}}}$ .

Our goal is to maximize the total probability assigned to any correct simple labeling; therefore, for each sentence/verb pair  $(s, v)$ , we want to increase

$\sum_{c_k^{sv} \in K^{sv}} P(c_k^{sv})$ . This expression treats the simple labeling (consisting of a simple sentence and a role assignment) as a hidden variable that is summed out. Taking the log, summing across all sentence/verb pairs, and adding L2 regularization on the weights, we have our final objective  $F(\mathbf{w})$ :

$$\sum_{s,v} \left( \log \frac{\sum_{c_k^{sv} \in K^{sv}} e^{\mathbf{w}^T \mathbf{f}_k^{sv}}}{\sum_{c_{k'}^{sv} \in C^{sv}} e^{\mathbf{w}^T \mathbf{f}_{k'}^{sv}}} \right) - \frac{\mathbf{w}^T \mathbf{w}}{2\sigma^2}$$

We train our model by optimizing the objective using standard methods, specifically BFGS. Due to the summation over the hidden variable representing the choice of simple sentence (not observed in the training data), our objective is not convex. Thus, we are not guaranteed to find a global optimum; in practice we have gotten good results using the default initialization of setting all weights to 0.

Consider the derivative of the likelihood component with respect to a single weight  $w_l$ :

$$\sum_{c_k^{sv} \in K^{sv}} \mathbf{f}_k^{sv}(l) \frac{P(c_k^{sv})}{\sum_{c_{k'}^{sv} \in K^{sv}} P(c_{k'}^{sv})} - \sum_{c_{k'}^{sv} \in C^{sv}} \mathbf{f}_{k'}^{sv}(l) P(c_{k'}^{sv})$$

where  $\mathbf{f}_k^{sv}(l)$  denotes the  $l^{\text{th}}$  component of  $\mathbf{f}_k^{sv}$ . This formula is positive when the expected value of the  $l^{\text{th}}$  feature is higher on the set of correct simple labelings  $K^{sv}$  than on the set of all simple labelings  $C^{sv}$ . Thus, the optimization procedure will tend to be self-reinforcing, increasing the score of correct simple labelings which already have a high score.

## 7 Simplification Data Structure

Our representation of the set of possible simplifications of a sentence addresses two computational bottlenecks. The first is the need to repeatedly copy large chunks of the sentence. For example, if we are depassivizing a sentence, we can avoid copying the subject and object of the original sentence by simply referring back to them in the depassivized version. At worst, we only need to add one node for each numbered node in the transformation rule. The second issue is the possible exponential blowup of the number of generated sentences. Consider “I want to eat and I want to drink and I want to play and ...” Each subsentence can be simplified, yielding two possibilities for each subsentence. The number of simplifications of the entire sentence is then exponential in the length of the sentence. However,



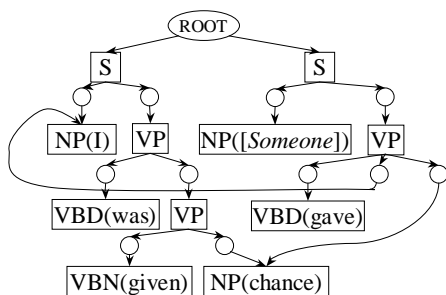


Figure 7: Data structure after applying the depassivize rule to “I was given (a) chance.” Circular nodes are OR-nodes, rectangular nodes are AND-nodes.

we can store these simplifications compactly as a set of independent decisions, “I {want to eat OR eat} and I {want to drink OR drink} and ...”

Both issues can be addressed by representing the set of simplifications using an AND-OR tree, a general data structure also used to store parse forests such as those produced by a chart parser. In our case, the AND nodes are similar to constituent nodes in a parse tree – each has a category (e.g. NP) and (if it is a leaf) a word (e.g. “chance”), but instead of having a list of child constituents, it instead has a list of child OR nodes. Each OR node has one or more constituent children that correspond to the different options at this point in the tree. Figure 7 shows the resulting AND-OR tree after applying the depassivize rule to the original parse of “I was given a chance.” Because this AND-OR tree represents only two different parses, the original parse and the depassivized version, only one OR node in the tree has more than one child – the root node, which has two choices, one for each parse. However, the AND nodes immediately above “I” and “chance” each have more than one OR-node parent, since they are shared by the original and depassivized parses<sup>1</sup>. To extract a parse from this data structure, we apply the following recursive algorithm: starting at the root OR node, each time we reach an OR node, we choose and recurse on exactly one of its children; each time we reach an AND node, we recurse on all of its children. In Figure 7, we have only one choice: if we go left at the root, we generate the original parse; otherwise, we generate the depassivized version.

Unfortunately, it is difficult to find the optimal AND-OR tree. We use a greedy but smart proce-

<sup>1</sup>In this particular example, both of these nodes are leaves, but in general shared nodes can be entire tree fragments

dure to try to produce a small tree. We omit details for lack of space. Using our rule set, the compact representation is usually (but not always) small.

For our compact representation to be useful, we need to be able to optimize our objective without expanding all possible simple sentences. A relatively straight-forward extension of the inside-outside algorithm for chart-parses allows us to learn and perform inference in our compact representation (a similar algorithm is presented in (Geman & Johnson, 2002)). We omit details for lack of space.

## 8 Experiments

We evaluated our system using the setup of the Conll 2005 semantic role labeling task.<sup>2</sup> Thus, we trained on Sections 2-21 of PropBank and used Section 24 as development data. Our test data includes both the selected portion of Section 23 of PropBank, plus the extra data on the Brown corpus. We used the Charniak parses provided by the Conll distribution.

We compared to a strong **Baseline** SRL system that learns a logistic regression model using the features of Pradhan et al. (2005). It has two stages. The first filters out nodes that are unlikely to be arguments. The second stage labels each remaining node either as a particular role (e.g. “ARGO”) or as a non-argument. Note that the baseline feature set includes a feature corresponding to the subcategorization of the verb (specifically, the sequence of nonterminals which are children of the predicate’s parent node). Thus, **Baseline** does have access to something similar to our model’s role pattern feature, although the **Baseline** subcategorization feature only includes post-verbal modifiers and is generally much noisier because it operates on the original sentence.

Our **Transforms** model takes as input the Charniak parses supplied by the Conll release, and labels every node with Core arguments (ARG0-ARG5). Our rule set does not currently handle either referent arguments (such as “who” in “The man who ate ...”) or non-core arguments (such as ARGM-TMP). For these arguments, we simply filled in using our baseline system (specifically, any non-core argument which did not overlap an argument predicted by our model was added to the labeling). Also, on some sentences, our system did not generate any predictions because no valid simple sen-

<sup>2</sup><http://www.lsi.upc.es/srlconll/home.html>

Model	Dev	Test WSJ	Test Brown	Test WSJ+Br
<b>Baseline</b>	74.7	76.9	64.7	75.3
<b>Transforms</b>	75.6	77.4	<b>66.8</b>	76.0
<b>Combined</b>	<b>76.0</b>	<b>78.0</b>	66.4	<b>76.5</b>
Punyakanok	77.35	79.44	67.75	77.92

Table 2: F1 Measure using Charniak parses

tences were produced by the simplification system . Again, we used the baseline to fill in predictions (for all arguments) for these sentences.

**Baseline** and **Transforms** were regularized using a Gaussian prior; for both models,  $\sigma^2 = 1.0$  gave the best results on the development set.

For generating role predictions from our model, we have two reasonable options: use the labeling given by the single highest scoring simple labeling; or compute the distribution over predictions for each node by summing over all simple labelings. The latter method worked slightly better, particularly when combined with the baseline model as described below, so all reported results use this method.

We also evaluated a hybrid model that combines the **Baseline** with our simplification model. For a given sentence/verb pair  $(s, v)$ , we find the set of constituents  $N^{sv}$  that made it past the first (filtering) stage of **Baseline**. For each candidate simple sentence/labeling pair  $c_k^{sv} = (t_i^{sv}, g_j^v)$  proposed by our model, we check to see which of the constituents in  $N^{sv}$  are already present in our simple sentence  $t_i^{sv}$ . Any constituents that are *not* present are then assigned a probability distribution over possible roles according to **Baseline**. Thus, we fall back **Baseline** whenever the current simple sentence does not have an “opinion” about the role of a particular constituent. The **Combined** model is thus able to correctly label sentences when the simplification process drops some of the arguments (generally due to unusual syntax). Each of the two components was trained separately and combined only at testing time.

Table 2 shows results of these three systems on the Conll-2005 task, plus the top-performing system (Punyakanok et al., 2005) for reference. **Baseline** already achieves good performance on this task, placing at about 75<sup>th</sup> percentile among evaluated systems. Our **Transforms** model outperforms **Baseline** on all sets. Finally, our **Combined** model improves over **Transforms** on all but the test Brown corpus,

Model	Test WSJ
<b>Baseline</b>	87.6
<b>Transforms</b>	88.2
<b>Combined</b>	<b>88.5</b>

Table 3: F1 Measure using gold parses

achieving a statistically significant increase over the **Baseline** system (according to confidence intervals calculated for the Conll-2005 results).

The **Combined** model still does not achieve the performance levels of the top several systems. However, these systems all use information from multiple parses, allowing them to fix many errors caused by incorrect parses. We return to this issue in Section 10. Indeed, as shown in Table 3, performance on gold standard parses is (as expected) much better than on automatically generated parses, for all systems. Importantly, the **Combined** model again achieves a significant improvement over **Baseline**.

We expect that by labeling simple sentences, our model will generalize well even on verbs with a small number of training examples. Figure 8 shows F1 measure on the WSJ test set as a function of training set size. Indeed, both the **Transforms** model and the **Combined** model significantly outperform the **Baseline** model when there are fewer than 20 training examples for the verb. While the **Baseline** model has higher accuracy than the **Transforms** model for verbs with a very large number of training examples, the **Combined** model is at or above both of the other models in all but the rightmost bucket, suggesting that it gets the best of both worlds.

We also found, as expected, that our model improved on sentences with very long parse paths. For example, in the sentence “Big investment banks refused to step up to the plate to support the beleaguered floor traders by buying blocks of stock, traders say,” the parse path from “buy” to its ARG0, “Big investment banks,” is quite long. The **Transforms** model correctly labels the arguments of “buy”, while the **Baseline** system misses the ARG0.

To understand the importance of different types of rules, we performed an ablation analysis. For each major rule category in Figure 1, we deleted those rules from the rule set, retrained, and evaluated using the **Combined** model. To avoid parse-related issues, we trained and evaluated on gold-standard parses. Most important were rules relating to (ba-

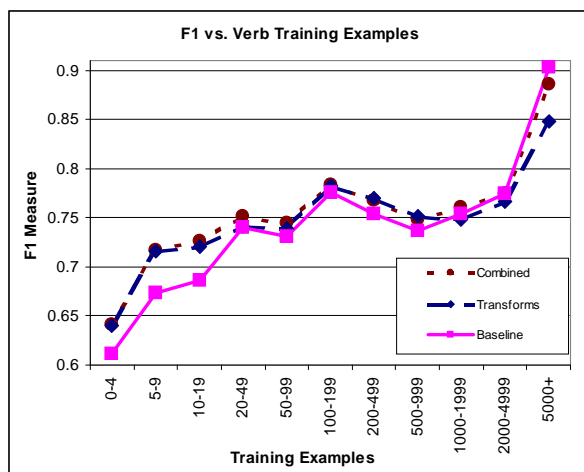


Figure 8: F1 Measure on the WSJ test set as a function of training set size. Each bucket on the X-axis corresponds to a group of verbs for which the number of training examples fell into the appropriate range; the value is the average performance for verbs in that bucket.

sic) verb raising/control, “make” rewrites, modified nouns, and passive constructions. Each of these rule categories when removed lowered the F1 score by approximately .4%. In contrast, removing rules for non-basic control, possessives, and inverted sentences caused a negligible reduction in performance. This may be because the relevant syntactic structures occur rarely; because **Baseline** does well on those constructs; or because the simplification model has trouble learning when to apply these rules.

## 9 Related Work

One area of current research which has similarities with this work is on Lexical Functional Grammars (LFGs). Both approaches attempt to abstract away from the surface level syntax of the sentence (e.g., the XLE system<sup>3</sup>). The most obvious difference between the approaches is that we use SRL data to train our system, avoiding the need to have labeled data specific to our simplification scheme.

There have been a number of works which model verb subcategorization. Approaches include incorporating a subcategorization feature (Gildea & Jurafsky, 2002; Xue & Palmer, 2004), such as the one used in our baseline; and building a model which jointly classifies all arguments of a verb (Toutanova et al., 2005). Our method differs from past work in that it extracts its role pattern feature from the simplified sentence. As a result, the feature is less noisy

<sup>3</sup><http://www2.parc.com/isl/groups/nltt/xle/>

and generalizes better across syntactic variation than a feature extracted from the original sentence.

Another group of related work focuses on summarizing sentences through a series of deletions (Jing, 2000; Dorr et al., 2003; Galley & McKeown, 2007). In particular, the latter two works iteratively simplify the sentence by deleting a phrase at a time. We differ from these works in several important ways. First, our transformation language is not context-free; it can reorder constituents and then apply transformation rules to the reordered sentence. Second, we are focusing on a somewhat different task; these works are interested in obtaining a single summary of each sentence which maintains all “essential” information, while in our work we produce a simplification that may lose semantic content, but aims to contain all arguments of a verb. Finally, training our model on SRL data allows us to avoid the relative scarcity of parallel simplification corpora and the issue of determining what is “essential” in a sentence.

Another area of related work in the semantic role labeling literature is that on tree kernels (Moschitti, 2004; Zhang et al., 2007). Like our method, tree kernels decompose the parse path into smaller pieces for classification. Our model can generalize better across verbs because it first simplifies, then classifies based on the simplified sentence. Also, through iterative simplifications we can discover structure that is not immediately apparent in the original parse.

## 10 Future Work

There are a number of improvements that could be made to the current simplification system, including augmenting the rule set to handle more constructions and doing further sentence normalizations, e.g., identifying whether a direct object exists. Another interesting extension involves incorporating parser uncertainty into the model; in particular, our simplification system is capable of seamlessly accepting a parse forest as input.

There are a variety of other tasks for which sentence simplification might be useful, including summarization, information retrieval, information extraction, machine translation and semantic entailment. In each area, we could either use the simplification system as learned on SRL data, or retrain the simplification model to maximize performance on the particular task.

## References

- Dorr, B., Zajic, D., & Schwartz, R. (2003). Hedge: A parse-and-trim approach to headline generation. *Proceedings of the HLT-NAACL Text Summarization Workshop and Document Understanding Conference*.
- Galley, M., & McKeown, K. (2007). Lexicalized markov grammars for sentence compression. *Proceedings of NAACL-HLT*.
- Geman, S., & Johnson, M. (2002). Dynamic programming for parsing and estimation of stochastic unification-based grammars. *Proceedings of ACL*.
- Gildea, D., & Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*.
- Jing, H. (2000). Sentence reduction for automatic text summarization. *Proceedings of Applied NLP*.
- Moschitti, A. (2004). A study on convolution kernels for shallow semantic parsing. *Proceedings of ACL*.
- Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J. H., & Jurafsky, D. (2005). Support vector learning for semantic argument classification. *Machine Learning*, 60, 11–39.
- Punyakanok, V., Koomen, P., Roth, D., & Yih, W. (2005). Generalized inference with multiple semantic role labeling systems. *Proceedings of CoNLL*.
- Toutanova, K., Haghghi, A., & Manning, C. (2005). Joint learning improves semantic role labeling. *Proceedings of ACL*, 589–596.
- Xue, N., & Palmer, M. (2004). Calibrating features for semantic role labeling. *Proceedings of EMNLP*.
- Zhang, M., Che, W., Aw, A., Tan, C. L., Zhou, G., Liu, T., & Li, S. (2007). A grammar-driven convolution tree kernel for semantic role classification. *Proceedings of ACL*.

# Summarizing Emails with Conversational Cohesion and Subjectivity

Giuseppe Carenini, Raymond T. Ng and Xiaodong Zhou

Department of Computer Science

University of British Columbia

Vancouver, BC, Canada

{carenini, rng, xdzhou}@cs.ubc.ca

## Abstract

In this paper, we study the problem of summarizing email conversations. We first build a sentence quotation graph that captures the conversation structure among emails. We adopt three cohesion measures: clue words, semantic similarity and cosine similarity as the weight of the edges. Second, we use two graph-based summarization approaches, Generalized ClueWordSummarizer and Page-Rank, to extract sentences as summaries. Third, we propose a summarization approach based on subjective opinions and integrate it with the graph-based ones. The empirical evaluation shows that the basic clue words have the highest accuracy among the three cohesion measures. Moreover, subjective words can significantly improve accuracy.

## 1 Introduction

With the ever increasing popularity of emails, it is very common nowadays that people discuss specific issues, events or tasks among a group of people by emails (Fisher and Moody, 2002). Those discussions can be viewed as conversations via emails and are valuable for the user as a personal information repository (Ducheneaut and Bellotti, 2001). In this paper, we study the problem of *summarizing email conversations*. Solutions to this problem can help users access the information embedded in emails more effectively. For instance, 10 minutes before a meeting, a user may want to quickly go through a previous discussion via emails that is going to be discussed soon. In that case, rather than

reading each individual email one by one, it would be preferable to read a concise summary of the previous discussion with the major information summarized. Email summarization is also helpful for mobile email users on a small screen.

Summarizing email conversations is challenging due to the characteristics of emails, especially the conversational nature. Most of the existing methods dealing with email conversations use the email *thread* to represent the email conversation structure, which is not accurate in many cases (Yeh and Harnly, 2006). Meanwhile, most existing email summarization approaches use quantitative features to describe the conversation structure, e.g., number of recipients and responses, and apply some general multi-document summarization methods to extract some sentences as the summary (Rambow et al., 2004) (Wan and McKeown, 2004). Although such methods consider the conversation structure somehow, they simplify the conversation structure into several features and do not fully utilize it into the summarization process.

In contrast, in this paper, we propose new summarization approaches by sentence extraction, which rely on a fine-grain representation of the conversation structure. We first build a *sentence quotation graph* by content analysis. This graph not only captures the conversation structure more accurately, especially for selective quotations, but it also represents the conversation structure at the finer granularity of sentences. As a second contribution of this paper, we study several ways to measure the cohesion between parent and child sentences in the quotation graph: *clue words* (re-occurring words in the reply)

(Carenini et al., 2007), *semantic similarity* and *cosine similarity*. Hence, we can directly evaluate the importance of each sentence in terms of its cohesion with related ones in the graph. The extractive summarization problem can be viewed as a node ranking problem. We apply two summarization algorithms, Generalized ClueWordSummarizer and Page-Rank to rank nodes in the sentence quotation graph and to select the corresponding most highly ranked sentences as the summary.

Subjective opinions are often critical in many conversations. As a third contribution of this paper, we study how to make use of the subjective opinions expressed in emails to support the summarization task. We integrate our best cohesion measure together with the subjective opinions. Our empirical evaluations show that subjective words and phrases can significantly improve email summarization.

To summarize, this paper is organized as follows. In Section 2, we discuss related work. After building a sentence quotation graph to represent the conversation structure in Section 3, we apply two summarization methods in Section 4. In Section 5, we study summarization approaches with subjective opinions. Section 6 presents the empirical evaluation of our methods. We conclude this paper and propose future work in Section 7.

## 2 Related Work

Rambow et al. proposed a sentence extraction summarization approach for email threads (Rambow et al., 2004). They described each sentence in an email conversations by a set of features and used machine learning to classify whether or not a sentence should be included into the summary. Their experiments showed that features about emails and the email thread could significantly improve the accuracy of summarization.

Wan et al. proposed a summarization approach for decision-making email discussions (Wan and McKeown, 2004). They extracted the issue and response sentences from an email thread as a summary. Similar to the issue-response relationship, Shrestha et al. (Shrestha and McKeown, 2004) proposed methods to identify the question-answer pairs from an email thread. Once again, their results showed that including features about the email

thread could greatly improve the accuracy. Similar results were obtained by Corston-Oliver et al. They studied how to identify “action” sentences in email messages and use those sentences as a summary (Corston-Oliver et al., 2004). All these approaches used the email thread as a coarse representation of the underlying conversation structure.

In our recent study (Carenini et al., 2007), we built a fragment quotation graph to represent an email conversation and developed a ClueWordSummarizer (CWS) based on the concept of clue words. Our experiments showed that CWS had a higher accuracy than the email summarization approach in (Rambow et al., 2004) and the generic multi-document summarization approach MEAD (Radev et al., 2004). Though effective, the CWS method still suffers from the following four substantial limitations. First, we used a fragment quotation graph to represent the conversation, which has a coarser granularity than the sentence level. For email summarization by sentence extraction, the fragment granularity may be inadequate. Second, we only adopted one cohesion measure (clue words that are based on stemming), and did not consider more sophisticated ones such as semantically similar words. Third, we did not consider subjective opinions. Finally, we did not compare CWS to other possible graph-based approaches as we propose in this paper.

Other than for email summarization, other document summarization methods have adopted graph-ranking algorithms for summarization, e.g., (Wan et al., 2007), (Mihalcea and Tarau, 2004) and (Erkan and Radev, 2004). Those methods built a complete graph for all sentences in one or multiple documents and measure the similarity between every pair of sentences. Graph-ranking algorithms, e.g., Page-Rank (Brin and Page, 1998), are then applied to rank those sentences. Our method is different from them. First, instead of using the complete graph, we build the graph based on the conversation structure. Second, we try various ways to compute the similarity among sentences and the ranking of the sentences.

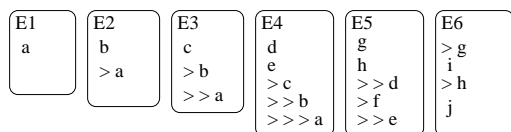
Several studies in the NLP literature have explored the reoccurrence of similar words within one document due to text cohesion. The idea has been formalized in the construct of *lexical chains* (Barzilay and Elhadad, 1997). While our approach and lexical chains both rely on lexical cohesion, they are

quite different with respect to the kind of linkages considered. Lexical chain is only based on similarities between lexical items in contiguous sentences. In contrast, in our approach, the linkage is based on the existing conversation structure. In our approach, the “chain” is not only “lexical” but also “conversational”, and typically spans over several emails.

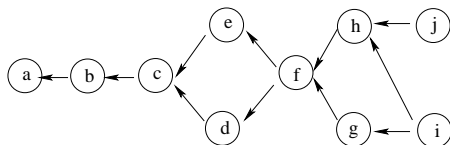
### 3 Extracting Conversations from Multiple Emails

In this section, we first review how to build a fragment quotation graph through an example. Then we extend this structure into a sentence quotation graph, which can allow us to capture the conversational relationship at the level of sentences.

#### 3.1 Building the Fragment Quotation Graph



(a) Conversation involving 6 Emails



(b) Fragment Quotation Graph

Figure 1: A Real Example

Figure 1(a) shows a real example of a conversation from a benchmark data set involving 6 emails. For the ease of representation, we do not show the original content but abbreviate them as a sequence of fragments. In the first step, all new and quoted fragments are identified. For instance, email  $E_3$  is decomposed into 3 fragments: new fragment  $c$  and quoted fragments  $b$ , which in turn quoted  $a$ .  $E_4$  is decomposed into  $de$ ,  $c$ ,  $b$  and  $a$ . Then, in the second step, to identify distinct fragments (nodes), fragments are compared with each other and overlaps are identified. Fragments are split if necessary (e.g., fragment  $gh$  in  $E_5$  is split into  $g$  and  $h$  when matched with  $E_6$ ), and duplicates are removed. At the end, 10 distinct fragments  $a, \dots, j$  give rise to 10 nodes in the graph shown in Figure 1(b).

As the third step, we create edges, which represent the replying relationship among fragments. In

general, it is difficult to determine whether one fragment is actually replying to another fragment. We assume that *any new fragment is a potential reply to neighboring quotations – quoted fragments immediately preceding or following it*. Let us consider  $E_6$  in Figure 1(a). there are two edges from node  $i$  to  $g$  and  $h$ , while there is only a single edge from  $j$  to  $h$ . For  $E_3$ , there are the edges  $(c, b)$  and  $(c, a)$ . Because of the edge  $(b, a)$ , the edge  $(c, a)$  is not included in Figure 1(b). Figure 1(b) shows the fragment quotation graph of the conversation shown in Figure 1(a) with all the redundant edges removed. In contrast, if threading is done at the coarse granularity of entire emails, as adopted in many studies, the threading would be a simple chain from  $E_6$  to  $E_5$ ,  $E_5$  to  $E_4$  and so on. Fragment  $f$  reflects a special and important phenomenon, where the original email of a quotation does not exist in the user’s folder. We call this as the *hidden email* problem. This problem and its influence on email summarization were studied in (Carenini et al., 2005) and (Carenini et al., 2007).

#### 3.2 Building the Sentence Quotation Graph

A fragment quotation graph can only represent the conversation in the fragment granularity. We notice that some sentences in a fragment are more relevant to the conversation than the remaining ones. The fragment quotation graph is not capable of representing this difference. Hence, in the following, we describe how to build a sentence quotation graph from the fragment quotation graph and introduce several ways to give weight to the edges.

In a sentence quotation graph  $GS$ , each node represents a distinct sentence in the email conversation, and each edge  $(u, v)$  represents the replying relationship between node  $u$  and  $v$ . The algorithm to create the sentence quotation graph contains the following 3 steps: create nodes, create edges and assign weight to edges. In the following, we first illustrate how to create nodes and edges. In Section 3.3, we discuss different ways to assign weight to edges.

Given a fragment quotation graph  $GF$ , we first split each fragment into a set of sentences. For each sentence, we create a node in the sentence quotation graph  $GS$ . In this way, each sentence in the email conversation is represented by a distinct node in  $GS$ .

As the second step, we create the edges in  $GS$ . The edges in  $GS$  are based on the edges in  $GF$

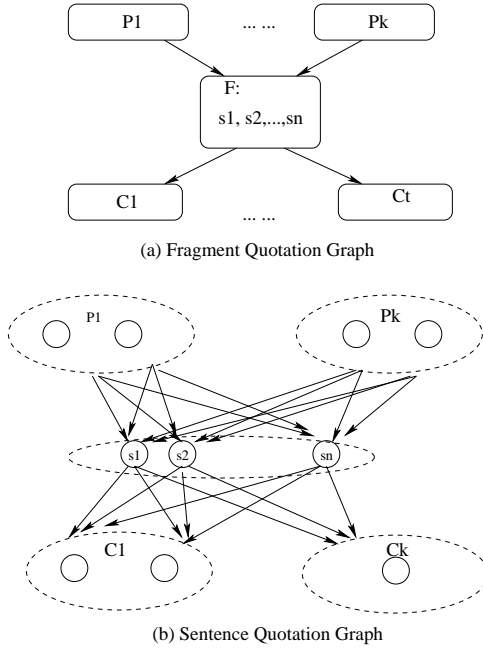


Figure 2: Create the Sentence Quotation Graph from the Fragment Quotation Graph

because the edges in  $GF$  already reflect the replying relationship among fragments. For each edge  $(u, v) \in GF$ , we create edges from each sentence of  $u$  to each sentence of  $v$  in the sentence quotation graph  $GS$ . This is illustrated in Figure 2.

Note that when each distinct sentence in an email conversation is represented as one node in the sentence quotation graph, the extractive email summarization problem is transformed into a standard node ranking problem within the sentence quotation graph. Hence, general node ranking algorithms, e.g., Page-Rank, can be used for email summarization as well.

### 3.3 Measuring the Cohesion Between Sentences

After creating the nodes and edges in the sentence quotation graph, a key technical question is how to measure the degree that two sentences are related to each other, e.g., a sentence  $s_u$  is replying to or being replied by  $s_v$ . In this paper, we use text cohesion between two sentences  $s_u$  and  $s_v$  to make this assessment and assign this as the weight of the corresponding edge  $(s_u, s_v)$ . We explore three types of cohesion measures: (1) clue words that are based on stems, (2) semantic distance based on WordNet

and (3) cosine similarity that is based on the word TFIDF vector. In the following, we discuss these three methods separately in detail.

#### 3.3.1 Clue Words

Clue words were originally defined as re-occurring words with the same stem between two adjacent fragments in the fragment quotation graph. In this section, we re-define clue words based on the sentence quotation graph as follows. A clue word in a sentence  $S$  is a non-stop word that also appears (modulo stemming) in a parent or a child node (sentence) of  $S$  in the sentence quotation graph.

The frequency of clue words in the two sentences measures their cohesion as described in Equation 1.

$$weight(s_u, s_v) = \sum_{w_i \in s_u} freq(w_i, s_v) \quad (1)$$

#### 3.3.2 Semantic Similarity Based on WordNet

Other than stems, when people reply to previous messages they may also choose some semantically related words, such as synonyms and antonyms, e.g., “talk” vs. “discuss”. Based on this observation, we propose to use semantic similarity to measure the cohesion between two sentences. We use the well-known lexical database WordNet to get the semantic similarity of two words. Specifically, we use the package by (Pedersen et al., 2004), which includes several methods to compute the semantic similarity. Among those methods, we choose “lesk” and “jcn”, which are considered two of the best methods in (Jurafsky and Martin, 2008). Similar to the clue words, we measure the semantic similarity of two sentences by the total semantic similarity of the words in both sentences. This is described in the following equation.

$$weight(s_u, s_v) = \sum_{w_i \in s_u} \sum_{w_j \in s_v} \sigma(w_i, w_j), \quad (2)$$

#### 3.3.3 Cosine Similarity

Cosine similarity is a popular metric to compute the similarity of two text units. To do so, each sentence is represented as a word vector of TFIDF values. Hence, the cosine similarity of two sentences  $s_u$  and  $s_v$  is then computed as  $\frac{\vec{s}_u \cdot \vec{s}_v}{\|\vec{s}_u\| \cdot \|\vec{s}_v\|}$ .



## 4 Summarization Based on the Sentence Quotation Graph

Having built the sentence quotation graph with different measures of cohesion, in this section, we develop two summarization approaches. One is the generalization of the CWS algorithm in (Carenini et al., 2007) and one is the well-known Page-Rank algorithm. Both algorithms compute a score,  $SentScore(s)$ , for each sentence (node)  $s$ , which is used to select the top- $k\%$  sentences as the summary.

### 4.1 Generalized ClueWordSummarizer

Given the sentence quotation graph, since the weight of an edge  $(s, t)$  represents the extent that  $s$  is related to  $t$ , a natural assumption is that the more relevant a sentence (node)  $s$  is to its parents and children, the more important  $s$  is. Based on this assumption, we compute the weight of a node  $s$  by summing up the weight of all the outgoing and incoming edges of  $s$ . This is described in the following equation.

$$SentScore(s) = \sum_{(s,t) \in GS} weight(s,t) + \sum_{(p,s) \in GS} weight(p,s) \quad (3)$$

The weight of an edge  $(s, t)$  can be any of the three metrics described in the previous section. Particularly, when the weight of the edge is based on clue words as in Equation 1, this method is equivalent to Algorithm CWS in (Carenini et al., 2007). In the rest of this paper, let CWS denote the Generalized ClueWordSummarizer when the edge weight is based on clue words, and let *CWS-Cosine* and *CWS-Semantic* denote the summarizer when the edge weight is cosine similarity and semantic similarity respectively. *Semantic* can be either “lesk” or “jcn”.

### 4.2 Page-Rank-based Summarization

The Generalized ClueWordSummarizer only considers the weight of the edges without considering the importance (weight) of the nodes. This might be incorrect in some cases. For example, a sentence replied by an important sentence should get some of its importance. This intuition is similar to the one inspiring the well-known Page-Rank algorithm. The traditional Page-Rank algorithm only considers the outgoing edges. In email conversations, what we want to measure is the cohesion between sentences no matter which one is being replied to. Hence, we

need to consider both incoming and outgoing edges and the corresponding sentences.

Given the sentence quotation graph  $G_s$ , the Page-Rank-based algorithm is described in Equation 4.  $PR(s)$  is the Page-Rank score of a node (sentence)  $s$ .  $d$  is the dumping factor, which is initialized to 0.85 as suggested in the Page-Rank algorithm. In this way, the rank of a sentence is evaluated globally based on the graph.

## 5 Summarization with Subjective Opinions

Other than the conversation structure, the measures of cohesion and the graph-based summarization methods we have proposed, the importance of a sentence in emails can be captured from other aspects. In many applications, it has been shown that sentences with subjective meanings are paid more attention than factual ones (Pang and Lee, 2004) (Esuli and Sebastiani, 2006). We evaluate whether this is also the case in emails, especially when the conversation is about decision making, giving advice, providing feedbacks, etc.

A large amount of work has been done on determining the level of subjectivity of text (Shanahan et al., 2005). In this paper we follow a very simple approach that, if successful, could be extended in future work. More specifically, in order to assess the degree of subjectivity of a sentence  $s$ , we count the frequency of words and phrases in  $s$  that are likely to bear subjective opinions. The assumption is that the more subjective words  $s$  contains, the more likely that  $s$  is an important sentence for the purpose of email summarization. Let  $SubjScore(s)$  denote the number of words with a subjective meaning. Equation 5 illustrates how  $SubjScore(s)$  is computed.  $SubjList$  is a list of words and phrases that indicate subjective opinions.

$$SubjScore(s) = \sum_{w_i \in SubjList, w_i \in s} freq(w_i) \quad (5)$$

The  $SubjScore(s)$  alone can be used to evaluate the importance of a sentence. In addition, we can combine  $SubjScore$  with any of the sentence scores based on the sentence quotation graph. In this paper, we use a simple approach by adding them up as the final sentence score.

$$PR(s) = (1 - d) + d * \frac{\sum_{s_i \in \text{child}(s)} \text{weight}(s, s_i) * PR(s_i) + \sum_{s_j \in \text{parent}(s)} \text{weight}(s_j, s) * PR(s_j)}{\sum_{s_i \in \text{child}(s)} \text{weight}(s, s_i) + \sum_{s_j \in \text{parent}(s)} \text{weight}(s_j, s)} \quad (4)$$

As to the subjective words and phrases, we consider the following two lists generated by researchers in this area.

- *OpFind*: The list of subjective words in (Wilson et al., 2005). The major source of this list is from (Riloff and Wiebe, 2003) with additional words from other sources. This list contains 8,220 words or phrases in total.
- *OpBear*: The list of opinion bearing words in (Kim and Hovy, 2005). This list contains 27,193 words or phrases in total.

## 6 Empirical Evaluation

### 6.1 Dataset Setup

There are no publicly available annotated corpora to test email summarization techniques. So, the first step in our evaluation was to develop our own corpus. We use the Enron email dataset, which is the largest public email dataset. In the 10 largest *inbox* folders in the Enron dataset, there are 296 email conversations. Since we are studying summarizing email conversations, we required that each selected conversation contained at least 4 emails. In total, 39 conversations satisfied this requirement. We use the MEAD package to segment the text into 1,394 sentences (Radev et al., 2004).

We recruited 50 human summarizers to review those 39 selected email conversations. Each email conversation was reviewed by 5 different human summarizers. For each given email conversation, human summarizers were asked to generate a summary by directly selecting important sentences from the original emails in that conversation. We asked the human summarizers to select 30% of the total sentences in their summaries.

Moreover, human summarizers were asked to classify each selected sentence as either *essential* or *optional*. The essential sentences are crucial to the email conversation and have to be extracted in any case. The optional sentences are not critical but

are useful to help readers understand the email conversation if the given summary length permits. By classifying essential and optional sentences, we can distinguish the core information from the supporting ones and find the most convincing sentences that most human summarizers agree on.

As essential sentences are more important than the optional ones, we give more weight to the essential selections. We compute a *GSValue* for each sentence to evaluate its importance according to the human summarizers' selections. The score is designed as follows: for each sentence  $s$ , one essential selection has a score of 3, one optional selection has a score of 1. Thus, the *GSValue* of a sentence ranges from 0 to 15 (5 human summarizers  $\times$  3). The *GSValue* of 8 corresponds to 2 essential and 2 optional selections. If a sentence has a *GSValue* no less than 8, we take it as an *overall essential* sentence. In the 39 conversations, we have about 12% overall essential sentences.

### 6.2 Evaluation Metrics

Evaluation of summarization is believed to be a difficult problem in general. In this paper, we use two metrics to measure the accuracy of a system generated summary. One is *sentence pyramid precision*, and the other is *ROUGE recall*. As to the statistical significance, we use the 2-tail pairwise student t-test in all the experiments to compare two specific methods. We also use ANOVA to compare three or more approaches together.

The sentence pyramid precision is a relative precision based on the *GSValue*. Since this idea is borrowed from the pyramid metric by Nenkova et al. (Nenkova et al., 2007), we call it the *sentence pyramid precision*. In this paper, we simplify it as the *pyramid precision*. As we have discussed above, with the reviewers' selections, we get a *GSValue* for each sentence, which ranges from 0 to 15. With this *GSValue*, we rank all sentences in a descendant order. We also group all sentences with the same *GSValue* together as one tier  $T_i$ , where  $i$  is the corre-

sponding GSValue;  $i$  is called the *level* of the tier  $T_i$ . In this way, we organize all sentences into a pyramid: a sequence of tiers with a descendant order of levels. With the pyramid of sentences, the accuracy of a summary is evaluated over the best summary we can achieve under the same summary length. The best summary of  $k$  sentences are the top  $k$  sentences in terms of GSValue.

Other than the sentence pyramid precision, we also adopt the ROUGE recall to evaluate the generated summary with a finer granularity than sentences, e.g., n-gram and longest common subsequence. Unlike the pyramid method which gives more weight to sentences with a higher GSValue, ROUGE is not sensitive to the difference between essential and optional selections (it considers all sentences in one summary equally). Directly applying ROUGE may not be accurate in our experiments. Hence, we use the overall essential sentences as the gold standard summary for each conversation, i.e., sentences in tiers no lower than  $T_8$ . In this way, the ROUGE metric measures the similarity of a system generated summary to a gold standard summary that is considered important by most human summarizers. Specifically, we choose ROUGE-2 and ROUGE-L as the evaluation metric.

### 6.3 Evaluating the Weight of Edges

In Section 3.3, we developed three ways to compute the weight of an edge in the sentence quotation graph, i.e., clue words, semantic similarity based on WordNet and cosine similarity. In this section, we compare them together to see which one is the best. It is well-known that the accuracy of the summarization method is affected by the length of the summary. In the following experiments, we choose the summary length as 10%, 12%, 15%, 20% and 30% of the total sentences and use the aggregated average accuracy to evaluate different algorithms.

Table 1 shows the aggregated pyramid precision over all five summary lengths of CWS, CWS-Cosine, two semantic similarities, i.e., CWS-lesk and CWS-jcn. We first use ANOVA to compare the four methods. For the pyramid precision, the  $F$  ratio is 50, and the p-value is  $2.1E-29$ . This shows that the four methods are significantly different in the average accuracy. In Table 1, by comparing CWS with the other methods, we can see that CWS obtains the

	CWS	CWS-Cosine	CWS-lesk	CWS-jcn
Pyramid	0.60	0.39	0.57	0.57
p-value		<0.0001	0.02	0.005
ROUGE-2	0.46	0.31	0.39	0.35
p-value		<0.0001	<0.001	<0.001
ROUGE-L	0.54	0.43	0.49	0.45
p-value		<0.0001	<0.001	<0.001

Table 1: Generalized CWS with Different Edge Weights

highest precision (0.60). The widely used cosine similarity does not perform well. Its precision (0.39) is about half of the precision of CWS with a p-value less than 0.0001. This clearly shows that CWS is significantly better than CWS-Cosine. Meanwhile, both semantic similarities have lower accuracy than CWS, and the differences are also statistically significant even with the conservative Bonferroni adjustment (i.e., the p-values in Table 1 are multiplied by three).

The above experiments show that the widely used cosine similarity and the more sophisticated semantic similarity in WordNet are less accurate than the basic CWS in the summarization framework. This is an interesting result and can be viewed at least from the following two aspects. First, clue words, though straight forward, are good at capturing the important sentences within an email conversation. The higher accuracy of CWS may suggest that people tend to use the same words to communicate in email conversations. Some related words in the previous emails are adopted exactly or in another similar format (modulo stemming). This is different from other documents such as newspaper articles and formal reports. In those cases, the authors are usually professional in writing and choose their words carefully, even intentionally avoid repeating the same words to gain some diversity. However, for email conversation summarization, this does not appear to be the case.

Moreover, in the previous discussion we only considered the accuracy in precision without considering the runtime issue. In order to have an idea of the runtime of the two methods, we did the following comparison. We randomly picked 1000 pairs of words from the 20 conversations and compute their semantic distance in “jcn”. It takes about 0.056 seconds to get the semantic similarity for one pair on the

average. In contrast, when the weight of edges are computed based on clue words, the average runtime to compute the SentScore for all sentences in a conversation is only 0.05 seconds, which is even a little less than the time to compute the semantic similarity of one pair of words. In other words, when CWS has generated the summary of one conversation, we can only get the semantic distance between one pair of words. Note that for each edge in the sentence quotation graph, we need to compute the distance for every pair of words in each sentence. Hence, the empirical results do not support the use of semantic similarity. In addition, we do not discuss the runtime performance of CWS-cosine here because of its extremely low accuracy.

#### 6.4 Comparing Page-Rank and CWS

Table 2 compares Page-Rank and CWS under different edge weights. We compare Page-Rank only with CWS because CWS is better than the other Generalized CWS methods as shown in the previous section. This table shows that Page-Rank has a lower accuracy than that of CWS and the difference is significant in all four cases. Moreover, when we compare Table 1 and 2 together, we can find that, for each kind of edge weight, Page-Rank has a lower accuracy than the corresponding Generalized CWS. Note that Page-Rank computes a node’s rank based on all the nodes and edges in the graph. In contrast, CWS only considers the similarity between neighboring nodes. The experimental result indicates that for email conversation, the local similarity based on clue words is more consistent with the human summarizers’ selections.

#### 6.5 Evaluating Subjective Opinions

Table 3 shows the result of using subjective opinions described in Section 5. The first 3 columns in this table are pyramid precision of CWS and using 2 lists of subjective words and phrases alone. We can see that by using subjective words alone, the precision of each subjective list is lower than that of CWS. However, when we integrate CWS and subjective words together, as shown in the remaining 2 columns, the precisions get improved consistently for both lists. The increase in precision is at least 0.04 with statistical significance. A natural question to ask is whether clue words and subjective words overlap much. Our

	CWS	PR-Clue	PR-Cosine	PR-lesk	PR-jcn
Pyramid	0.60	0.51	0.37	0.54	0.50
p-value		< 0.0001	< 0.0001	< 0.0001	< 0.0001
ROUGE-2	0.46	0.4	0.26	0.36	0.39
p-value		0.05	< 0.0001	0.001	0.02
ROUGE-L	0.54	0.49	0.36	0.44	0.48
p-value		0.06	< 0.0001	0.0005	0.02

Table 2: Compare Page-Rank with CWS

	CWS	OpFind	OpBear	CWS+OpFind	CWS+OpBear
Pyramid	0.60	0.52	0.59	<b>0.65</b>	<b>0.64</b>
p-value		0.0003	0.8	<0.0001	0.0007
ROUGE-2	0.46	0.37	0.44	<b>0.50</b>	<b>0.49</b>
p-value		0.0004	0.5	0.004	0.06
ROUGE-L	0.54	0.48	0.56	<b>0.60</b>	<b>0.59</b>
p-value		0.01	0.6	0.0002	0.002

Table 3: Accuracy of Using Subjective Opinions

analysis shows that the overlap is minimal. For the list of OpFind, the overlapped words are about 8% of clue words and 4% of OpFind that appear in the conversations. This result clearly shows that clue words and subjective words capture the importance of sentences from different angles and can be used together to gain a better accuracy.

## 7 Conclusions

We study how to summarize email conversations based on the conversational cohesion and the subjective opinions. We first create a sentence quotation graph to represent the conversation structure on the sentence level. We adopt three cohesion metrics, clue words, semantic similarity and cosine similarity, to measure the weight of the edges. The Generalized ClueWordSummarizer and Page-Rank are applied to this graph to produce summaries. Moreover, we study how to include subjective opinions to help identify important sentences for summarization.

The empirical evaluation shows the following two discoveries: (1) The basic CWS (based on clue words) obtains a higher accuracy and a better runtime performance than the other cohesion measures. It also has a significant higher accuracy than the Page-Rank algorithm. (2) By integrating clue words and subjective words (phrases), the accuracy of CWS is improved significantly. This reveals an interesting phenomenon and will be further studied.

## References

Regina Barzilay and Michael Elhadad. 1997. Using lexical chains for text summarization. In *Proceedings of*

- the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL, Madrid, Spain.*
- 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.
- Giuseppe Carenini, Raymond T. Ng, and Xiaodong Zhou. 2005. Scalable discovery of hidden emails from large folders. In *ACM SIGKDD'05*, pages 544–549.
- Giuseppe Carenini, Raymond T. Ng, and Xiaodong Zhou. 2007. Summarizing email conversations with clue words. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 91–100.
- Simon Corston-Oliver, Eric K. Ringger, Michael Gamon, and Richard Campbell. 2004. Integration of email and task lists. In *First conference on email and anti-spam (CEAS)*, Mountain View, California, USA, July 30–31.
- Nicolas Ducheneaut and Victoria Bellotti. 2001. E-mail as habitat: an exploration of embedded personal information management. *Interactions*, 8(5):30–38.
- Günes Erkan and Dragomir R. Radev. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22:457–479.
- Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation*, May 24–26.
- Danyel Fisher and Paul Moody. 2002. Studies of automated collection of email records. In *University of Irvine ISR Technical Report UCI-ISR-02-4*.
- Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (Second Edition)*. Prentice-Hall.
- Soo-Min Kim and Eduard Hovy. 2005. Automatic detection of opinion bearing words and sentences. In *Proceedings of the Second International Joint Conference on Natural Language Processing: Companion Volume*, Jeju Island, Republic of Korea, October 11–13.
- R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, July.
- Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: incorporating human content selection variation in summarization evaluation. *ACM Transaction on Speech and Language Processing*, 4(2):4.
- Bo Pang and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 271–278.
- Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proceedings of Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-04)*, pages 38–41, May 3–5.
- Dragomir R. Radev, Hongyan Jing, Malgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40(6):919–938, November.
- Owen Rambow, Lokesh Shrestha, John Chen, and Chirsty Lauridsen. 2004. Summarizing email threads. In *HLT/NAACL*, May 2–7.
- Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*, pages 105–112.
- James G. Shanahan, Yan Qu, and Janyce Wiebe. 2005. *Computing Attitude and Affect in Text: Theory and Applications (The Information Retrieval Series)*. Springer-Verlag New York, Inc.
- Lokesh Shrestha and Kathleen McKeown. 2004. Detection of question-answer pairs in email conversations. In *Proceedings of COLING'04*, pages 889–895, August 23–27.
- Stephen Wan and Kathleen McKeown. 2004. Generating overview summaries of ongoing email thread discussions. In *Proceedings of COLING'04, the 20th International Conference on Computational Linguistics*, August 23–27.
- XiaoJun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 552–559, Prague, Czech Republic, June.
- Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Opinionfinder: a system for subjectivity analysis. In *Proceedings of HLT/EMNLP on Interactive Demonstrations*, pages 34–35.
- Jen-Yuan Yeh and Aaron Harnly. 2006. Email thread reassembly using similarity matching. In *Third Conference on Email and Anti-Spam (CEAS)*, July 27 - 28.

# Ad Hoc Treebank Structures

**Markus Dickinson**

Department of Linguistics

Indiana University

md7@indiana.edu

## Abstract

We outline the problem of ad hoc rules in treebanks, rules used for specific constructions in one data set and unlikely to be used again. These include ungeneralizable rules, erroneous rules, rules for ungrammatical text, and rules which are not consistent with the rest of the annotation scheme. Based on a simple notion of rule equivalence and on the idea of finding rules unlike any others, we develop two methods for detecting ad hoc rules in flat treebanks and show they are successful in detecting such rules. This is done by examining evidence across the grammar and without making any reference to context.

## 1 Introduction and Motivation

When extracting rules from constituency-based treebanks employing flat structures, grammars often limit the set of rules (e.g., Charniak, 1996), due to the large number of rules (Krotov et al., 1998) and “leaky” rules that can lead to mis-analysis (Foth and Menzel, 2006). Although frequency-based criteria are often used, these are not without problems because low-frequency rules can be valid and potentially useful rules (see, e.g., Daelemans et al., 1999), and high-frequency rules can be erroneous (see, e.g., Dickinson and Meurers, 2005). A key issue in determining the rule set is rule generalizability: will these rules be needed to analyze new data? This issue is of even more importance when considering the task of porting a parser trained on one genre to another genre (e.g., Gildea, 2001). Infrequent rules in one genre may be quite frequent in

another (Sekine, 1997) and their frequency may be unrelated to their usefulness for parsing (Foth and Menzel, 2006). Thus, we need to carefully consider the applicability of rules in a treebank to new text.

Specifically, we need to examine *ad hoc* rules, rules used for particular constructions specific to one data set and unlikely to be used on new data. This is why low-frequency rules often do not extend to new data: if they were only used once, it was likely for a specific reason, not something we would expect to see again. Ungeneralizable rules, however, do not extend to new text for a variety of reasons, not all of which can be captured strictly by frequency.

While there are simply phenomena which, for various reasons, are rarely used (e.g., long coordinated lists), other ungeneralizable phenomena are potentially more troubling. For example, when ungrammatical or non-standard text is used, treebanks employ rules to cover it, but do not usually indicate ungrammaticality in the annotation. These rules are only to be used in certain situations, e.g., for typographical conventions such as footnotes, and the fact that the situation is irregular would be useful to know if the purpose of an induced grammar is to support robust parsing. And these rules are outright damaging if the set of treebank rules is intended to accurately capture the grammar of a language. This is true of precision grammars, where analyses can be more or less preferred (see, e.g., Wagner et al., 2007), and in applications like intelligent computer-aided language learning, where learner input is parsed to detect what is correct or not (see, e.g., Vandeventer Faltin, 2003, ch. 2). If a treebank grammar is used (e.g., Metcalf and Boyd,

2006), then one needs to isolate rules for ungrammatical data, to be able to distinguish grammatical from ungrammatical input.

Detecting ad hoc rules can also reveal issues related to rule quality. Many ad hoc rules exist because they are erroneous. Not only are errors inherently undesirable for obtaining an accurate grammar, but training on data with erroneous rules can be detrimental to parsing performance (e.g., Dickinson and Meurers, 2005; Hogan, 2007). As annotation schemes are not guaranteed to be completely consistent, other ad hoc rules point to non-uniform aspects of the annotation scheme. Thus, identifying ad hoc rules can also provide feedback on annotation schemes, an especially important step if one is to use the treebank for specific applications (see, e.g., Vadas and Curran, 2007), or if one is in the process of developing a treebank.

Although statistical techniques have been employed to detect anomalous annotation (Ule and Simov, 2004; Eskin, 2000), these methods do not account for linguistically-motivated generalizations across rules, and no full evaluation has been done on a treebank. Our starting point for detecting ad hoc rules is also that they are dissimilar to the rest of the grammar, but we rely on a notion of equivalence which accounts for linguistic generalizations, as described in section 2. We generalize equivalence in a corpus-independent way in section 3 to detect ad hoc rules, using two different methods to determine when rules are dissimilar. The results in section 4 show the success of the method in identifying all types of ad hoc rules.

## 2 Background

### 2.1 Equivalence classes

To define dissimilarity, we need a notion of similarity, and, a starting point for this is the error detection method outlined in Dickinson and Meurers (2005). Since most natural language expressions are endocentric, i.e., a category projects to a phrase of the same category (e.g., X-bar Schema, Jackendoff, 1977), daughters lists with more than one possible mother are flagged as potentially containing an error. For example, IN NP<sup>1</sup> has nine different mothers in the Wall Street Journal (WSJ) portion of the Penn

<sup>1</sup>Appendix A lists all categories used in this paper.

Treebank (Marcus et al., 1993), six of which are errors.

This method can be extended to increase recall, by treating similar daughters lists as equivalent (Dickinson, 2006, 2008). For example, the daughters lists ADVP RB ADVP and ADVP , RB ADVP in (1) can be put into the same equivalence class, because they predict the same mother category. With this equivalence, the two different mothers, PP and ADVP, point to an error (in PP).

- (1) a. to slash its work force in the U.S. , [*PP* [*ADVP* as] soon/*RB* [*ADVP* as next month]]
- b. to report ... [*ADVP* [*ADVP* immediately] ,/ not/*RB* [*ADVP* a month later]]

Anything not contributing to predicting the mother is ignored in order to form equivalence classes. Following the steps below, 15,989 daughters lists are grouped into 3783 classes in the WSJ.

1. Remove daughter categories that are always non-predictive to phrase categorization, i.e., always adjuncts, such as punctuation and the parenthetical (PRN) category.
2. Group head-equivalent lexical categories, e.g., NN (common noun) and NNS (plural noun).
3. Model adjacent identical elements as a single element, e.g., NN NN becomes NN.

While the sets of non-predictive and head-equivalent categories are treebank-specific, they require only a small amount of manual effort.

### 2.2 Non-equivalence classes

Rules in the same equivalence class not only predict the same mother, they provide support that the daughters list is accurate—the more rules within a class, the better evidence that the annotation scheme legitimately licenses that sequence. A lack of similar rules indicates a potentially anomalous structure.

Of the 3783 equivalence classes for the whole WSJ, 2141 are unique, i.e., have only one unique daughters list. For example, in (2), the daughters list RB TO JJ NNS is a daughters list with no correlates in the treebank; it is erroneous because *close to wholesale* needs another layer of structure, namely adjective phrase (ADJP) (Bies et al., 1995, p. 179).

(2) they sell [merchandise] for [ $NP$  close/RB to/TO wholesale/JJ prices/NNS ]

Using this strict equivalence to identify ad hoc rules is quite successful (Dickinson, 2008), but it misses a significant number of generalizations. These equivalences were not designed to assist in determining linguistic patterns from non-linguistic patterns, but to predict the mother category, and thus many correct rules are incorrectly flagged. To provide support for the correct rule  $NP \rightarrow DT CD JJS NNP JJ NNS$  in (3), for instance, we need to look at some highly similar rules in the treebank, e.g., the three instances of  $NP \rightarrow DT CD JJ NNP NNS$ , which are not strictly equivalent to the rule in (3).

(3) [ $NP$  the/DT 100/CD largest/JJS Nasdaq/NNP financial/JJ stocks/NNS ]

### 3 Rule dissimilarity and generalizability

#### 3.1 Criteria for rule equivalence

With a notion of (non-)equivalence as a heuristic, we can begin to detect ad hoc rules. First, however, we need to redefine equivalence to better reflect syntactic patterns.

Firstly, in order for two rules to be in the same equivalence class—or even to be similar—the mother must also be the same. This captures the property that identical daughters lists with different mothers are distinct (cf. Dickinson and Meurers, 2005). For example, looking back at (1), the one occurrence of  $ADVP \rightarrow ADVP, RB ADVP$  is very similar to the 4 instances of  $ADVP \rightarrow RB ADVP$ , whereas the one instance of  $PP \rightarrow ADVP RB ADVP$  is not and is erroneous. Daughters lists are thus now only compared to rules with the same mother.

Secondly, we use only two steps to determine equivalence: 1) remove non-predictive daughter categories, and 2) group head-equivalent lexical categories.<sup>2</sup> While useful for predicting the same mother, the step of Kleene reduction is less useful for our purposes since it ignores potential differences in argument structure. It is important to know how many identical categories can appear within a given rule, to tell whether it is reliable;  $VP \rightarrow VB$

$NP$  and  $VP \rightarrow VB NP NP$ , for example, are two different rules.<sup>3</sup>

Thirdly, we base our scores on token counts, in order to capture the fact that the more often we observe a rule, the more reliable it seems to be. This is not entirely true, as mentioned above, but this prevents frequent rules such as  $NP \rightarrow EX$  (1075 occurrences) from being seen as an anomaly.

With this new notion of equivalence, we can now proceed to accounting for similar rules in detecting ad hoc rules.

#### 3.2 Reliability scores

In order to devise a scoring method to reflect similar rules, the simplest way is to use a version of edit distance between rules, as we do under the *Whole daughters scoring* below. This reflects the intuition that rules with similar lists of daughters reflect the same properties. This is the “positive” way of scoring rules, in that we start with a basic notion of equivalence and look for more positive evidence that the rule is legitimate. Rules without such evidence are likely ad hoc.

Our goal, though, is to take the results and examine the anomalous rules, i.e., those which lack strong evidence from other rules. We can thus more directly look for “negative” evidence that a rule is ad hoc. To do this, we can examine the weakest parts of each rule and compare those across the corpus, to see which anomalous patterns emerge; we do this in the *Bigram scoring* section below.

Because these methods exploit different properties of rules and use different levels of abstraction, they have complementary aspects. Both start with the same assumptions about what makes rules equivalent, but diverge in how they look for rules which do not fit well into these equivalences.

**Whole daughters scoring** The first method to detect ad hoc rules directly accounts for similar rules across equivalence classes. Each rule type is assigned a reliability score, calculated as follows:

1. Map a rule to its equivalence class.
2. For every rule token within the equivalence class, add a score of 1.

<sup>2</sup>See Dickinson (2006) for the full mappings.

<sup>3</sup>Experiments done with Kleene reduction show that the results are indeed worse.



3. For every rule token within a highly similar equivalence class, add a score of  $\frac{1}{2}$ .

Positive evidence that a rule is legitimate is obtained by looking at similar classes in step #3, and then rules with the lowest scores are flagged as potentially ad hoc (see section 4.1). To determine similarity, we use a modified Levenshtein distance, where only insertions and deletions are allowed; a distance of one qualifies as *highly similar*.<sup>4</sup> Allowing two or more changes would be problematic for unary rules (e.g., (4a), and in general, would allow us to add and subtract dissimilar categories. We thus remain conservative in determining similarity.

Also, we do not utilize substitutions: while they might be useful in some cases, it is too problematic to include them, given the difference in meaning of each category. Consider the problematic rules in (4). In (4a), which occurs once, if we allow substitutions, then we will find 760 “comparable” instances of VP → VB, despite the vast difference in category (verb vs. adverb). Likewise, the rule in (4b), which occurs 8 times, would be “comparable” to the 602 instances of PP → IN PP, used for multi-word prepositions like *because of*.<sup>5</sup> To maintain these true differences, substitutions are not allowed.

- (4) a. VP → RB  
 b. PP → JJ PP

This notion of similarity captures many generalizations, e.g., that adverbial phrases are optional. For example, in (5), the rule reduces to S → PP ADVP NP ADVP VP. With a strict notion of equivalence, there are no comparable rules. However, the class S → PP NP ADVP VP, with 198 members, is highly similar, indicating more confidence in this correct rule.

- (5) [<sub>S</sub> [<sub>PP</sub> During his years in Chiriqui] ,/ , [<sub>ADVP</sub> however] ,/ , [<sub>NP</sub> Mr. Noriega] [<sub>ADVP</sub> also] [<sub>VP</sub> revealed himself as an officer as perverse as he was ingenious] ./ .]

<sup>4</sup>The score is thus more generally  $\frac{1}{1+distance}$ , although we ascribe no theoretical meaning to this

<sup>5</sup>Rules like PP → JJ PP might seem to be correct, but this depends upon the annotation scheme. Phrases starting with *due to* are sometimes annotated with this rule, but they also occur as ADJP or ADVP or with *due* as RB. If PP → JJ PP is correct, identifying this rule actually points to other erroneous rules.

**Bigram scoring** The other method of detecting ad hoc rules calculates reliability scores by focusing specifically on what the classes do not have in common. Instead of examining and comparing rules in their entirety, this method abstracts a rule to its component parts, similar to features using information about *n*-grams of daughter nodes in parse reranking models (e.g., Collins and Koo, 2005).

We abstract to bigrams, including added *START* and *END* tags, as longer sequences risk missing generalizations; e.g., unary rules would have no comparable rules. We score rule types as follows:

1. Map a rule to its equivalence class, resulting in a *reduced rule*.
2. Calculate the frequency of each <mother,bigram> pair in a reduced rule: for every reduced rule token with the same pair, add a score of 1 for that bigram pair.
3. Assign the score of the least-frequent bigram as the score of the rule.

We assign the score of the lowest-scoring bigram because we are interested in anomalous sequences. This is in the spirit of Květon and Oliva (2002), who define invalid bigrams for POS annotation sequences in order to detect annotation errors..

As one example, consider (6), where the reduced rule NP → NP DT NNP is composed of the bigrams START NP, NP DT, DT NNP, and NNP END. All of these are relatively common (more than a hundred occurrences each), except for NP DT, which appears in only two other rule types. Indeed, DT is an incorrect tag (NNP is correct): when NP is the first daughter of NP, it is generally a possessive, precluding the use of a determiner.

- (6) (NP (NP ABC 's) ( ' ' ' ' ) (DT This) (NNP Week))

The whole daughters scoring misses such problematic structures because it does not explicitly look for anomalies. The disadvantage of the bigram scoring, however, is its missing of the big picture: for example, the erroneous rule NP → NNP CC NP gets a large score (1905) because each subsequence is quite common. But this exact sequence is rather rare (NNP and NP are not generally coordinated), so the whole daughters scoring assigns a low score (4.0).

## 4 Evaluation

To gauge our success in detecting ad hoc rules, we evaluate the reliability scores in two main ways: 1) whether unreliable rules generalize to new data (section 4.1), and, more importantly, 2) whether the unreliable rules which do generalize are ad hoc in other ways—e.g., erroneous (section 4.2). To measure this, we use sections 02-21 of the WSJ corpus as training data to derive scores, section 23 as testing data, and section 24 as development data.

### 4.1 Ungeneralizable rules

To compare the effectiveness of the two scoring methods in identifying ungeneralizable rules, we examine how many rules from the training data do not appear in the heldout data, for different thresholds. In figure 1, for example, the method identifies 3548 rules with scores less than or equal to 50, 3439 of which do not appear in the development data, resulting in an ungeneralizability rate of 96.93%.

To interpret the figures below, we first need to know that of the 15,246 rules from the training data, 1832 occur in the development data, or only 12.02%, corresponding to 27,038 rule tokens. There are also 396 new rules in the development data, making for a total of 2228 rule types and 27,455 rule tokens.

#### 4.1.1 Development data results

The results are shown in figure 1 for the whole daughters scoring method and in figure 2 for the bigram method. Both methods successfully identify rules with little chance of occurring in new data, the whole daughters method performing slightly better.

Thresh.	Rules	Unused	Ungen.
1	311	311	100.00%
25	2683	2616	97.50%
50	3548	3439	96.93%
100	4596	4419	96.15%

Figure 1: Whole daughter ungeneralizability (devo.)

#### 4.1.2 Comparing across data

Is this ungeneralizability consistent over different data sets? To evaluate this, we use the whole daughters scoring method, since it had a higher ungeneralizability rate in the development data, and we use

Thresh.	Rules	Unused	Ungen.
1	599	592	98.83%
5	1661	1628	98.01%
10	2349	2289	97.44%
15	2749	2657	96.65%
20	3120	2997	96.06%

Figure 2: Bigram ungeneralizability (devo.)

section 23 of the WSJ and the Brown corpus portion of the Penn Treebank.

Given different data sizes, we now report the coverage of rules in the heldout data, for both type and token counts. For instance, in figure 3, for a threshold of 50, 108 rule types appear in the development data, and they appear 141 times. With 2228 total rule types and 27,455 rule tokens, this results in coverages of 4.85% and 0.51%, respectively.

In figures 3, 4, and 5, we observe the same trends for all data sets: low-scoring rules have little generalizability to new data. For a cutoff of 50, for example, rules at or below this mark account for approximately 5% of the rule types used in the data and half a percent of the tokens.

Thresh.	Types		Tokens	
	Used	Cov.	Used	Cov.
10	23	1.03%	25	0.09%
25	67	3.01%	78	0.28%
50	108	4.85%	141	0.51%
100	177	7.94%	263	0.96%
All	1832	82.22%	27,038	98.48%

Figure 3: Coverage of rules in WSJ, section 24

Thresh.	Types		Tokens	
	Used	Cov.	Used	Cov.
10	33	1.17%	39	0.08%
25	82	2.90%	117	0.25%
50	155	5.49%	241	0.51%
100	242	8.57%	416	0.88%
All	2266	80.24%	46,375	98.74%

Figure 4: Coverage of rules in WSJ, section 23

Note in the results for the larger Brown corpus that the percentage of overall rule types from the

Thresh.	Types		Tokens	
	Used	Cov.	Used	Cov.
10	187	1.51%	603	0.15%
25	402	3.25%	1838	0.45%
50	562	4.54%	2628	0.64%
100	778	6.28%	5355	1.30%
All	4675	37.75%	398,136	96.77%

Figure 5: Coverage of rules in Brown corpus

training data is only 37.75%, vastly smaller than the approximately 80% from either WSJ data set. This illustrates the variety of the grammar needed to parse this data versus the grammar used in training.

We have isolated thousands of rules with little chance of being observed in the evaluation data, and, as we will see in the next section, many of the rules which appear are problematic in other ways. The ungeneralizability results make sense, in light of the fact that reliability scores are based on token counts. Using reliability scores, however, has the advantage of being able to identify infrequent but correct rules (cf. example (5)) and also frequent but unhelpful rules. For example, in (7), we find erroneous cases from the development data of the rules  $WHNP \rightarrow WHNP WHPP$  (*five* should be NP) and  $VP \rightarrow NNP NP$  (*OKing* should be VBG). These rules appear 27 and 16 times, respectively, but have scores of only 28.0 and 30.5, showing their unreliability. Future work can separate the effect of frequency from the effect of similarity (see also section 4.3).

- (7) a. [ $WHNP$  [ $WHNP$  five] [ $WHPP$  of whom]]  
 b. received hefty sums for \* [ $VP$  OKing/NNP [ $NP$  the purchase of ...]]

## 4.2 Other ad hoc rules

The results in section 4.1 are perhaps unsurprising, given that many of the identified rules are simply rare. What is important, therefore, is to figure out why some rules appeared in the heldout data at all. As this requires qualitative analysis, we hand-examined the rules appearing in the development data. We set out to examine about 100 rules, and so we report only for the corresponding threshold, finding that ad hoc rules are predominant.

For the whole daughters scoring, at the 50 threshold, 55 (50.93%) of the 108 rules in the development

data are errors. Adding these to the ungeneralizable rules, 98.48% (3494/3548) of the 3548 rules are unhelpful for parsing, at least for this data set. An additional 12 rules cover non-English or fragmented constructions, making for 67 clearly ad hoc rules.

For the bigram scoring, at the 20 threshold, 67 (54.47%) of the 123 rules in the development data are erroneous, and 8 more are ungrammatical. This means that 97.88% (3054/3120) of the rules at this threshold are unhelpful for parsing this data, still slightly lower than the whole daughters scoring.

### 4.2.1 Problematic cases

But what about the remaining rules for both methods which are not erroneous or ungrammatical? First, as mentioned at the outset, there are several cases which reveal non-uniformity in the annotation scheme or guidelines. This may be justifiable, but it has an impact on grammars using the annotation scheme. Consider the case of NAC (not a constituent), used for complex NP premodifiers. The description for tagging titles in the guidelines (Bies et al., 1995, p. 208-209) covers the exact case found in section 24, shown in (8a). This rule,  $NAC \rightarrow NP PP$ , is one of the lowest-scoring rules which occurs, with a whole daughters score of 2.5 and a bigram score of 3, yet it is correct. Examining the guidelines more closely, however, we find examples such as (8b). Here, no extra NP layer is added, and it is not immediately clear what the criteria are for having an intermediate NP.

- (8) a. a “ [ $NAC$  [ $NP$  Points] [ $PP$  of Light]] ” foundation  
 b. The Wall Street Journal “ [ $NAC$  American Way [ $PP$  of Buying]] ” Survey

Secondly, rules with mothers which are simply rare are prone to receive lower scores, regardless of their generalizability. For example, the rules dominated by SINV, SQ, or SBARQ are all correct (6 in whole daughters, 5 in bigram), but questions are not very frequent in this news text: SQ appears only 350 times and SBARQ 222 times in the training data. One might thus consider normalizing the scores based on the overall frequency of the parent.

Finally, and most prominently, there are issues with coordinate structures. For example,  $NP \rightarrow NN CC DT$  receives a low whole daughters score of 7.0,

despite the fact that  $NP \rightarrow NN$  and  $NP \rightarrow DT$  are very common rules. This is a problem for both methods: for the whole daughters scoring, of the 108, 28 of them had a conjunct (CC or CONJP) in the daughters list, and 18 of these were correct. Likewise, for the bigram scoring, 18 had a conjunct, and 12 were correct. Reworking similarity scores to reflect coordinate structures and handle each case separately would require treebank-specific knowledge: the Penn Treebank, for instance, distinguishes unlike coordinated phrases (UCP) from other coordinated phrases, each behaving differently.

#### 4.2.2 Comparing the methods

There are other cases in which one method outperforms the other, highlighting their strengths and weaknesses. In general, both methods fare badly with clausal rules, i.e., those dominated by S, SBAR, SINV, SQ, or SBARQ, but the effect is slightly greater on the bigram scoring, where 20 of the 123 rules are clausal, and 16 of these are correct (i.e., 80% of them are misclassified). To understand this, we have to realize that most modifiers are adjoined at the sentence level when there is any doubt about their attachment (Bies et al., 1995, p. 13), leading to correct but rare subsequences. In sentence (9), for example, the reduced rule  $S \rightarrow SBAR PP NP VP$  arises because both the introductory SBAR and the PP are at the same level. This SBAR PP sequence is fairly rare, resulting in a bigram score of 13.

- (9) [ $S$  [ $SBAR$  As the best opportunities for corporate restructurings are exhausted \* of course] /, [ $PP$  at some point] [ $NP$  the market] [ $VP$  will start \* to reject them] /.]

Whole daughters scoring, on the other hand, assigns this rule a high reliability score of 2775.0, due to the fact that both SBAR NP VP and PP NP VP sequences are common. For rules with long modifier sequences, whole daughters scoring seems to be more effective since modifiers are easily skipped over in comparing to other rules. Whole daughters scoring is also imprecise with clausal rules (10/12 are misclassified), but identifies less of them, and they tend to be for rare mothers (see above).

Various cases are worse for the whole daughters scoring. First are quantifier phrases (QPs), which have a highly varied set of possible heads and argu-

ments. QP is “used for multiword numerical expressions that occur within NP (and sometimes ADJP), where the QP corresponds frequently to some kind of complex determiner phrase” (Bies et al., 1995, p. 193). This definition leads to rules which look different from QP to QP. Some of the lowest-scoring, correct rules are shown in (10). We can see that there is not a great deal of commonality about what comprises quantifier phrases, even if subparts are common and thus not flagged by the bigram method.

- (10) a. [ $QP$  only/RB three/CD of/IN the/DT nine/CD] justices  
 b. [ $QP$  too/RB many/JJ] cooks  
 c. 10 % [ $QP$  or/CC more/JJR]

Secondly, whole daughters scoring relies on complete sequences, and thus whether Kleene reduction (step #3 in section 2) is used makes a marked difference. For example, in (11), the rule  $NP \rightarrow DT JJ NNP NNP JJ NN NN$  is completely correct, despite its low whole daughters score of 15.5 and one occurrence. This rule is similar to the 10 occurrences of  $NP \rightarrow DT JJ NNP JJ NN$  in the training set, but we cannot see this without performing Kleene reduction. For noun phrases at least, using Kleene reduction might more accurately capture comparability. This is less of an issue for bigram scoring, as all the bigrams are perfectly valid, resulting here in a relatively high score (556).

- (11) [ $NP$  the/DT basic/JJ Macintosh/NNP Plus/NNP central/JJ processing/NN unit/NN ]

#### 4.3 Discriminating rare rules

In an effort to determine the effectiveness of the scores on isolating structures which are not linguistically sound, in a way which factors out frequency, we sampled 50 rules occurring only once in the training data. We marked for each whether it was correct or how it was ad hoc, and we did this blindly, i.e., without knowledge of the rule scores. Of these 50, only 9 are errors, 2 cover ungrammatical constructions, and 8 more are unclear. Looking at the bottom 25 scores, we find that the whole daughters and bigrams methods both find 6 errors, or 67% of them, additionally finding 5 unclear cases for the whole daughters and 6 for the bigrams method. Erroneous rules in the top half appear to be ones which

happened to be errors, but could actually be correct in other contexts (e.g., NP → NN NNP NNP CD). Although it is a small data set, the scores seem to be effectively sorting rare rules.

## 5 Summary and Outlook

We have outlined the problem of ad hoc rules in treebanks—ungeneralizable rules, erroneous rules, rules for ungrammatical text, and rules which are not necessarily consistent with the rest of the annotation scheme. Based on the idea of finding rules unlike any others, we have developed methods for detecting ad hoc rules in flat treebanks, simply by examining properties across the grammar and without making any reference to context.

We have been careful not to say how to use the reliability scores. First, without 100% accuracy, it is hard to know what their removal from a parsing model would mean. Secondly, assigning confidence scores to rules, as we have done, has a number of other potential applications. Parse reranking techniques, for instance, rely on knowledge about features other than those found in the core parsing model in order to determine the best parse (e.g., Collins and Koo, 2005; Charniak and Johnson, 2005). Active learning techniques also require a scoring function for parser confidence (e.g., Hwa et al., 2003), and often use uncertainty scores of parse trees in order to select representative samples for learning (e.g., Tang et al., 2002). Both could benefit from more information about rule reliability.

Given the success of the methods, we can strive to make them more corpus-independent, by removing the dependence on equivalence classes. In some ways, comparing rules to similar rules already naturally captures equivalences among rules. In this process, it will also be important to sort out the impact of similarity from the impact of frequency on identifying ad hoc structures.

## Acknowledgments

Thanks to the three anonymous reviewers for their helpful comments. This material is based upon work supported by the National Science Foundation under Grant No. IIS-0623837.

## A Relevant Penn Treebank categories

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
RB	Adverb
TO	<i>to</i>
VB	Verb, base form
VBG	Verb, gerund or present participle

Figure 6: POS tags in the PTB (Santorini, 1990)

ADJP	Adjective Phrase
ADVP	Adverb Phrase
CONJP	Conjunction Phrase
NAC	Not A Constituent
NP	Noun Phrase
PP	Prepositional Phrase
PRN	Parenthetical
QP	Quantifier Phrase
S	Simple declarative clause
SBAR	Clause introduced by subordinating conjunction
SBARQ	Direct question introduced by <i>wh</i> -word/phrase
SINV	Inverted declarative sentence
SQ	Inverted yes/no question
UCP	Unlike Coordinated Phrase
VP	Verb Phrase
WHNP	<i>Wh</i> -noun Phrase
WHPP	<i>Wh</i> -prepositional Phrase

Figure 7: Syntactic categories in the PTB (Bies et al., 1995)

## References

- Bies, Ann, Mark Ferguson, Karen Katz and Robert MacIntyre (1995). *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. University of Pennsylvania.
- Charniak, Eugene (1996). *Tree-Bank Grammars*. Tech. Rep. CS-96-02, Department of Computer Science, Brown University, Providence, RI.

- Charniak, Eugene and Mark Johnson (2005). Coarse-to-fine  $n$ -best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL-05*. Ann Arbor, MI, USA, pp. 173–180.
- Collins, Michael and Terry Koo (2005). Discriminative Reranking for Natural Language Parsing. *Computational Linguistics* 31(1), 25–69.
- Daelemans, Walter, Antal van den Bosch and Jakub Zavrel (1999). Forgetting Exceptions is Harmful in Language Learning. *Machine Learning* 34, 11–41.
- Dickinson, Markus (2006). Rule Equivalence for Error Detection. In *Proceedings of TLT 2006*. Prague, Czech Republic.
- Dickinson, Markus (2008). Similarity and Dissimilarity in Treebank Grammars. In *18th International Congress of Linguists (CIL18)*. Seoul.
- Dickinson, Markus 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 TLT 2005*. Barcelona, Spain.
- Eskin, Eleazar (2000). Automatic Corpus Correction with Anomaly Detection. In *Proceedings of NAACL-00*. Seattle, Washington, pp. 148–153.
- Foth, Kilian and Wolfgang Menzel (2006). Robust Parsing: More with Less. In *Proceedings of the workshop on Robust Methods in Analysis of Natural Language Data (ROMAND 2006)*.
- Gildea, Daniel (2001). Corpus Variation and Parser Performance. In *Proceedings of EMNLP-01*. Pittsburgh, PA.
- Hogan, Deirdre (2007). Coordinate Noun Phrase Disambiguation in a Generative Parsing Model. In *Proceedings of ACL-07*. Prague, pp. 680–687.
- Hwa, Rebecca, Miles Osborne, Anoop Sarkar and Mark Steedman (2003). Corrected Co-training for Statistical Parsers. In *Proceedings of ICML-2003*. Washington, DC.
- Jackendoff, Ray (1977). *X' Syntax: A Study of Phrase Structure*. Cambridge, MA: MIT Press.
- Krotov, Alexander, Mark Hepple, Robert J. Gaizauskas and Yorick Wilks (1998). Compacting the Penn Treebank Grammar. In *Proceedings of ACL-98*. pp. 699–703.
- Květon, Pavel and Karel Oliva (2002). Achieving an Almost Correct PoS-Tagged Corpus. In *Text, Speech and Dialogue (TSD)*. pp. 19–26.
- Marcus, M., Beatrice Santorini and M. A. Marcinkiewicz (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2), 313–330.
- Metcalf, Vanessa and Adriane Boyd (2006). Head-lexicalized PCFGs for Verb Subcategorization Error Diagnosis in ICALL. In *Workshop on Interfaces of Intelligent Computer-Assisted Language Learning*. Columbus, OH.
- Santorini, Beatrice (1990). *Part-Of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision, 2nd printing)*. Tech. Rep. MS-CIS-90-47, The University of Pennsylvania, Philadelphia, PA.
- Sekine, Satoshi (1997). The Domain Dependence of Parsing. In *Proceedings of ANLP-96*. Washington, DC.
- Tang, Min, Xiaoqiang Luo and Salim Roukos (2002). Active Learning for Statistical Natural Language Parsing. In *Proceedings of ACL-02*. Philadelphia, pp. 120–127.
- Ule, Tylman and Kiril Simov (2004). Unexpected Productions May Well be Errors. In *Proceedings of LREC 2004*. Lisbon, Portugal, pp. 1795–1798.
- Vadas, David and James Curran (2007). Adding Noun Phrase Structure to the Penn Treebank. In *Proceedings of ACL-07*. Prague, pp. 240–247.
- Vandeventer Faltin, Anne (2003). Syntactic error diagnosis in the context of computer assisted language learning. Thèse de doctorat, Université de Genève, Genève.
- Wagner, Joachim, Jennifer Foster and Josef van Genabith (2007). A Comparative Evaluation of Deep and Shallow Approaches to the Automatic Detection of Common Grammatical Errors. In *Proceedings of EMNLP-CoNLL 2007*. pp. 112–121.

# A Single Generative Model for Joint Morphological Segmentation and Syntactic Parsing

**Yoav Goldberg**

Ben Gurion University of the Negev  
Department of Computer Science  
POB 653 Be'er Sheva, 84105, Israel  
yoavg@cs.bgu.ac.il

**Reut Tsarfaty**

Institute for Logic Language and Computation  
University of Amsterdam  
Plantage Muidergracht 24, Amsterdam, NL  
rtsarfat@science.uva.nl

## Abstract

Morphological processes in Semitic languages deliver space-delimited words which introduce multiple, distinct, syntactic units into the structure of the input sentence. These words are in turn highly ambiguous, breaking the assumption underlying most parsers that the yield of a tree for a given sentence is known in advance. Here we propose a single joint model for performing both morphological segmentation and syntactic disambiguation which bypasses the associated circularity. Using a tree-bank grammar, a data-driven lexicon, and a linguistically motivated unknown-tokens handling technique our model outperforms previous pipelined, integrated or factorized systems for Hebrew morphological and syntactic processing, yielding an error reduction of 12% over the best published results so far.

## 1 Introduction

Current state-of-the-art broad-coverage parsers assume a direct correspondence between the lexical items ingrained in the proposed syntactic analyses (the *yields* of syntactic *parse-trees*) and the space-delimited tokens (henceforth, ‘tokens’) that constitute the unanalyzed surface forms (*utterances*). In Semitic languages the situation is very different.

In Modern Hebrew (Hebrew), a Semitic language with very rich morphology, particles marking conjunctions, prepositions, complementizers and relativizers are bound elements prefixed to the word (Glinert, 1989). The Hebrew token ‘*bcl*’<sup>1</sup>, for example, stands for the complete prepositional phrase

<sup>1</sup>We adopt here the transliteration of (Sima’an et al., 2001).

“in the shadow”. This token may further embed into a larger utterance, e.g., ‘*bcl hneim*’ (literally “in-the-shadow the-pleasant”, meaning roughly “in the pleasant shadow”) in which the dominated Noun is modified by a preceding space-delimited adjective. It should be clear from the onset that the particle *b* (“in”) in ‘*bcl*’ may then attach higher than the bare noun *cl* (“shadow”). This leads to word- and constituent-boundaries discrepancy, which breaks the assumptions underlying current state-of-the-art statistical parsers.

One way to approach this discrepancy is to assume a preceding phase of *morphological segmentation* for extracting the different lexical items that exist at the token level (as is done, to the best of our knowledge, in all parsing related work on Arabic and its dialects (Chiang et al., 2006)). The input for the segmentation task is however highly ambiguous for Semitic languages, and surface forms (tokens) may admit multiple possible analyses as in (Bar-Haim et al., 2007; Adler and Elhadad, 2006). The aforementioned surface form *bcl*, for example, may also stand for the lexical item “onion”, a Noun. The implication of this ambiguity for a parser is that the yield of syntactic trees no longer consists of space-delimited tokens, and the expected number of leaves in the syntactic analysis is not known in advance.

Tsarfaty (2006) argues that for Semitic languages determining the correct morphological segmentation is dependent on syntactic context and shows that increasing information sharing between the morphological and the syntactic components leads to improved performance on the joint task. Cohen and Smith (2007) followed up on these results and pro-

posed a system for joint inference of morphological and syntactic structures using factored models each designed and trained on its own.

Here we push the single-framework conjecture across the board and present a single model that performs morphological segmentation and syntactic disambiguation in a fully generative framework. We claim that no particular morphological segmentation is a-priori more likely for surface forms before exploring the compositional nature of syntactic structures, including manifestations of various long-distance dependencies. Morphological segmentation decisions in our model are delegated to a *lexeme-based* PCFG and we show that using a simple treebank grammar, a data-driven lexicon, and a linguistically motivated unknown-tokens handling our model outperforms (Tsarfaty, 2006) and (Cohen and Smith, 2007) on the joint task and achieves state-of-the-art results on a par with current respective standalone models.<sup>2</sup>

## 2 Modern Hebrew Structure

**Segmental morphology** Hebrew consists of seven particles *m* (“from”) *f* (“when”/“who”/“that”) *h* (“the”) *w* (“and”) *k* (“like”) *l* (“to”) and *b* (“in”). which may never appear in isolation and must always attach as prefixes to the following open-class category item we refer to as *stem*. Several such particles may be prefixed onto a single stem, in which case the affixation is subject to strict linear precedence constraints. Co-occurrences among the particles themselves are subject to further syntactic and lexical constraints relative to the stem.

While the linear precedence of segmental morphemes within a token is subject to constraints, the dominance relations among their mother and sister constituents is rather free. The relativizer *f* (“that”) for example, may attach to an arbitrarily long relative clause that goes beyond token boundaries. The attachment in such cases encompasses a long distance dependency that cannot be captured by Markovian processes that are typically used for morphological disambiguation. The same argument holds for resolving PP attachment of a prefixed preposition or marking conjunction of elements of any kind.

A less canonical representation of segmental mor-

phology is triggered by a morpho-phonological process of omitting the definite article *h* when occurring after the particles *b* or *l*. This process triggers ambiguity as for the definiteness status of Nouns following these particles. We refer to such cases in which the concatenation of elements does not strictly correspond to the original surface form as *super-segmental morphology*. An additional case of super-segmental morphology is the case of Pronominal Clitics. Inflectional features marking pronominal elements may be attached to different kinds of categories marking their pronominal complements. The additional morphological material in such cases appears *after* the stem and realizes the extended meaning. The current work treats both segmental and super-segmental phenomena, yet we note that there may be more adequate ways to treat super-segmental phenomena assuming Word-Based morphology as we explore in (Tsarfaty and Goldberg, 2008).

**Lexical and Morphological Ambiguity** The rich morphological processes for deriving Hebrew stems give rise to a high degree of ambiguity for Hebrew space-delimited tokens. The form *fmnh*, for example, can be understood as the verb “lubricated”, the possessed noun “her oil”, the adjective “fat” or the verb “got fat”. Furthermore, the systematic way in which particles are prefixed to one another and onto an open-class category gives rise to a distinct sort of morphological ambiguity: space-delimited tokens may be ambiguous between several different segmentation possibilities. The same form *fmnh* can be segmented as *f-mnh*, *f* (“that”) functioning as a relativizer with the form *mnh*. The form *mnh* itself can be read as at least three different verbs (“counted”, “appointed”, “was appointed”), a noun (“a portion”), and a possessed noun (“her kind”).

Such ambiguities cause discrepancies between token boundaries (indexed as white spaces) and constituent boundaries (imposed by syntactic categories) with respect to a surface form. Such discrepancies can be aligned via an intermediate level of PoS tags. PoS tags impose a unique morphological segmentation on surface tokens and present a unique valid yield for syntactic trees. The correct ambiguity resolution of the syntactic level therefore helps to resolve the morphological one, and vice versa.

<sup>2</sup>Standalone parsing models assume a segmentation Oracle.



### 3 Previous Work on Hebrew Processing

Morphological analyzers for Hebrew that analyze a surface form in isolation have been proposed by Segal (2000), Yona and Wintner (2005), and recently by the knowledge center for processing Hebrew (Itai et al., 2006). Such analyzers propose multiple segmentation possibilities and their corresponding analyses for a token in isolation but have no means to determine the most likely ones. Morphological disambiguators that consider a token in context (an utterance) and propose the most likely morphological analysis of an utterance (including segmentation) were presented by Bar-Haim et al. (2005), Adler and Elhadad (2006), Shacham and Wintner (2007), and achieved good results (the best segmentation result so far is around 98%).

The development of the very first Hebrew Treebank (Sima'an et al., 2001) called for the exploration of general statistical parsing methods, but the application was at first limited. Sima'an et al. (2001) presented parsing results for a DOP tree-gram model using a small data set (500 sentences) and semi-automatic morphological disambiguation. Tsarfaty (2006) was the first to demonstrate that fully automatic Hebrew parsing is feasible using the newly available 5000 sentences treebank. Tsarfaty and Sima'an (2007) have reported state-of-the-art results on Hebrew unlexicalized parsing (74.41%) albeit assuming oracle morphological segmentation.

The joint morphological and syntactic hypothesis was first discussed in (Tsarfaty, 2006; Tsarfaty and Sima'an, 2004) and empirically explored in (Tsarfaty, 2006). Tsarfaty (2006) used a morphological analyzer (Segal, 2000), a PoS tagger (Bar-Haim et al., 2005), and a general purpose parser (Schmid, 2000) in an integrated framework in which morphological and syntactic components interact to share information, leading to improved performance on the joint task. Cohen and Smith (2007) later on based a system for joint inference on factored, independent, morphological and syntactic components of which scores are combined to cater for the joint inference task. Both (Tsarfaty, 2006; Cohen and Smith, 2007) have shown that a single integrated framework outperforms a completely streamlined implementation, yet neither has shown a single generative model which handles both tasks.

### 4 Model Preliminaries

#### 4.1 The Status Space-Delimited Tokens

A Hebrew surface token may have several readings, each of which corresponding to a sequence of segments and their corresponding PoS tags. We refer to different readings as different *analyses* whereby the segments are deterministic given the sequence of PoS tags. We refer to a segment and its assigned PoS tag as a *lexeme*, and so analyses are in fact sequences of lexemes. For brevity we omit the segments from the analysis, and so analysis of the form “*f*/REL *mnh*/VB is represented simply as REL VB.

Such tag sequences are often treated as “complex tags” (e.g. REL+VB) (cf. (Bar-Haim et al., 2007; Habash and Rambow, 2005)) and probabilities are assigned to different analyses in accordance with the likelihood of their tags (e.g., “*f**mnh* is 30% likely to be tagged NN and 70% likely to be tagged REL+VB”). Here we do not submit to this view. When a token *f**mnh* is to be interpreted as the lexeme sequence *f*/REL *mnh*/VB, the analysis introduces two distinct entities, the relativizer *f* (“that”) and the verb *mnh* (“counted”), and not as the complex entity “that counted”. When the same token is to be interpreted as a single lexeme *f**mnh*, it may function as a single adjective “fat”. There is no relation between these two interpretations other than the fact that their surface forms coincide, and we argue that the only reason to prefer one analysis over the other is compositional. A possible probabilistic model for assigning probabilities to complex analyses of a surface form may be

$$P(\text{REL, VB}|f\text{mnh, context}) = P(\text{REL}|f)P(\text{VB}|mnh, \text{REL})P(\text{REL, VB}| \text{context})$$

and indeed recent sequential disambiguation models for Hebrew (Adler and Elhadad, 2006) and Arabic (Smith et al., 2005) present similar models.

We suggest that in unlexicalized PCFGs the syntactic context may be explicitly modeled in the derivation probabilities. Hence, we take the probability of the event *f**mnh* analyzed as REL VB to be

$$P(\text{REL} \rightarrow f|\text{REL}) \times P(\text{VB} \rightarrow mnh|\text{VB})$$

This means that we generate *f* and *mnh* independently depending on their corresponding PoS tags,

and the context (as well as the syntactic relation between the two) is modeled via the derivation resulting in a sequence REL VB spanning the form *fmnh*.

## 4.2 Lattice Representation

We represent all morphological analyses of a given utterance using a lattice structure. Each lattice arc corresponds to a segment and its corresponding PoS tag, and a path through the lattice corresponds to a specific morphological segmentation of the utterance. This is by now a fairly standard representation for multiple morphological segmentation of Hebrew utterances (Adler, 2001; Bar-Haim et al., 2005; Smith et al., 2005; Cohen and Smith, 2007; Adler, 2007). Figure 1 depicts the lattice for a 2-words sentence *bclm hneim*. We use double-circles to indicate the space-delimited token boundaries. Note that in our construction arcs can never cross token boundaries. Every token is independent of the others, and the sentence lattice is in fact a concatenation of smaller lattices, one for each token. Furthermore, some of the arcs represent lexemes not present in the input tokens (e.g. *h/DT*, *fl/POS*), however these are parts of valid analyses of the token (cf. *super-segmental morphology* section 2). Segments with the same surface form but different PoS tags are treated as different lexemes, and are represented as separate arcs (e.g. the two arcs labeled *neim* from node 6 to 7).

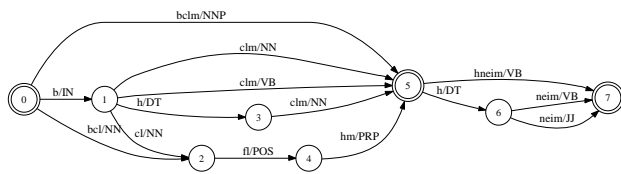


Figure 1: The Lattice for the Hebrew Phrase *bclm hneim*

A similar structure is used in speech recognition. There, a lattice is used to represent the possible sentences resulting from an interpretation of an acoustic model. In speech recognition the arcs of the lattice are typically weighted in order to indicate the probability of specific transitions. Given that weights on all outgoing arcs sum up to one, weights induce a probability distribution on the lattice paths. In sequential tagging models such as (Adler and Elhadad, 2006; Bar-Haim et al., 2007; Smith et al., 2005) weights are assigned according to a language model

based on linear context. In our model, however, all lattice paths are taken to be a-priori equally likely.

## 5 A Generative PCFG Model

The input for the joint task is a sequence  $W = w_1, \dots, w_n$  of space-delimited tokens. Each token may admit multiple analyses, each of which a sequence of one or more lexemes (we use  $l_i$  to denote a lexeme) belonging a presupposed Hebrew lexicon  $LEX$ . The entries in such a lexicon may be thought of as meaningful surface segments paired up with their PoS tags  $l_i = \langle s_i, p_i \rangle$ , but note that a surface segment  $s$  need not be a space-delimited token.

**The Input** The set of analyses for a token is thus represented as a lattice in which every arc corresponds to a specific lexeme  $l$ , as shown in Figure 1. A morphological analyzer  $M : \mathcal{W} \rightarrow \mathcal{L}$  is a function mapping sentences in Hebrew ( $W \in \mathcal{W}$ ) to their corresponding lattices ( $M(W) = L \in \mathcal{L}$ ). We define the lattice  $L$  to be the concatenation of the lattices  $L_i$  corresponding to the input words  $w_i$  (s.t.  $M(w_i) = L_i$ ). Each connected path  $\langle l_1 \dots l_k \rangle \in L$  corresponds to one morphological segmentation possibility of  $W$ .

**The Parser** Given a sequence of input tokens  $W = w_1 \dots w_n$  and a morphological analyzer, we look for the most probable parse tree  $\pi$  s.t.

$$\hat{\pi} = \arg \max_{\pi} P(\pi|W, M)$$

Since the lattice  $L$  for a given sentence  $W$  is determined by the morphological analyzer  $M$  we have

$$\hat{\pi} = \arg \max_{\pi} P(\pi|W, M, L)$$

Hence, our parser searches for a parse tree  $\pi$  over lexemes  $\langle l_1 \dots l_k \rangle$  s.t.  $l_i = \langle s_i, p_i \rangle \in LEX$ ,  $\langle l_1 \dots l_k \rangle \in L$  and  $M(W) = L$ . So we remain with

$$\hat{\pi} = \arg \max_{\pi} P(\pi|L)$$

which is precisely the formula corresponding to the so-called lattice parsing familiar from speech recognition. Every parse  $\pi$  selects a specific morphological segmentation  $\langle l_1 \dots l_k \rangle$  (a path through the lattice). This is akin to PoS tags sequences induced by different parses in the setup familiar from English and explored in e.g. (Charniak et al., 1996).

Our use of an unweighted lattice reflects our belief that all the segmentations of the given input sentence are a-priori equally likely; the only reason to prefer one segmentation over the another is due to the overall syntactic context which is modeled via the PCFG derivations. A compatible view is presented by Charniak et al. (1996) who consider the kind of probabilities a generative parser should get from a PoS tagger, and concludes that these should be  $P(w|t)$  “and nothing fancier”.<sup>3</sup> In our setting, therefore, the Lattice is *not* used to induce a probability distribution on a linear context, but rather, it is used as a common-denominator of state-indexation of all segmentations possibilities of a surface form. This is a unique object for which we are able to define a proper probability model. Thus our proposed model is a proper model assigning probability mass to all  $\langle \pi, L \rangle$  pairs, where  $\pi$  is a parse tree and  $L$  is the one and only lattice that a sequence of characters (and spaces)  $W$  over our alpha-beth gives rise to.

$$\sum_{\pi, L} P(\pi, L) = 1; L \text{ uniquely index } W$$

**The Grammar** Our parser looks for the most likely tree spanning a single path through the lattice of which the yield is a sequence of lexemes. This is done using a simple PCFG which is *lexeme-based*. This means that the rules in our grammar are of two kinds: (a) syntactic rules relating non-terminals to a sequence of non-terminals and/or PoS tags, and (b) lexical rules relating PoS tags to lattice arcs (lexemes). The possible analyses of a surface token pose constraints on the analyses of specific segments. In order to pass these constraints onto the parser, the lexical rules in the grammar are of the form  $p_i \rightarrow \langle s_i, p_i \rangle$

**Parameter Estimation** The grammar probabilities are estimated from the corpus using simple relative frequency estimates. Lexical rules are estimated in a similar manner. We smooth  $P_{rf}(p \rightarrow \langle s, p \rangle)$  for rare and OOV segments ( $s \in l, l \in L, s$  unseen) using a “per-tag” probability distribution over rare segments which we estimate using relative frequency estimates for once-occurring segments.

<sup>3</sup>An English sentence with ambiguous PoS assignment can be trivially represented as a lattice similar to our own, where every pair of consecutive nodes correspond to a word, and every possible PoS assignment for this word is a connecting arc.

**Handling Unknown tokens** When handling unknown *tokens* in a language such as Hebrew various important aspects have to be borne in mind. Firstly, Hebrew unknown tokens are doubly unknown: each unknown token may correspond to several segmentation possibilities, and each segment in such sequences may be able to admit multiple PoS tags. Secondly, some segments in a proposed segment sequence may in fact be *seen* lexical events, i.e., for some  $p$  tag  $P_{rf}(p \rightarrow \langle s, p \rangle) > 0$ , while other segments have never been observed as a lexical event before. The latter arcs correspond to OOV words in English. Finally, the assignments of PoS tags to OOV segments is subject to language specific constraints relative to the token it was originated from.

Our smoothing procedure takes into account all the aforementioned aspects and works as follows. We first make use of our morphological analyzer to find all segmentation possibilities by chopping off all prefix sequence possibilities (including the empty prefix) and construct a lattice off of them. The remaining arcs are marked OOV. At this stage the lattice path corresponds to segments only, with no PoS assigned to them. In turn we use two sorts of heuristics, orthogonal to one another, to prune segmentation possibilities based on *lexical* and *grammatical* constraints. We simulate *lexical* constraints by using an external lexical resource against which we verify whether OOV segments are in fact valid Hebrew lexemes. This heuristics is used to prune all segmentation possibilities involving “lexically improper” segments. For the remaining arcs, if the segment is in fact a known lexeme it is tagged as usual, but for the OOV arcs which are valid Hebrew entries lacking tags assignment, we assign all possible tags and then simulate a *grammatical* constraint. Here, all token-internal collocations of tags unseen in our training data are pruned away. From now on all lattice arcs are tagged segments and the assignment of probability  $P(p \rightarrow \langle s, p \rangle)$  to lattice arcs proceeds as usual.<sup>4</sup>

A rather pathological case is when our lexical heuristics prune away all segmentation possibilities and we remain with an empty lattice. In such cases we use the non-pruned lattice including all (possibly ungrammatical) segmentation, and let the statistics (including OOV) decide. We empirically control for

<sup>4</sup>Our heuristics may slightly alter  $\sum_{\pi, L} P(\pi, L) \approx 1$

the effect of our heuristics to make sure our pruning does not undermine the objectives of our joint task.

## 6 Experimental Setup

Previous work on morphological and syntactic disambiguation in Hebrew used different sets of data, different splits, differing annotation schemes, and different evaluation measures. Our experimental setup therefore is designed to serve two goals. Our primary goal is to exploit the resources that are most appropriate for the task at hand, and our secondary goal is to allow for comparison of our models’ performance against previously reported results. When a comparison against previous results requires additional pre-processing, we state it explicitly to allow for the reader to replicate the reported results.

**Data** We use the Hebrew Treebank, (Sima’an et al., 2001), provided by the knowledge center for processing Hebrew, in which sentences from the daily newspaper “Ha’aretz” are morphologically segmented and syntactically annotated. The treebank has two versions, v1.0 and v2.0, containing 5001 and 6501 sentences respectively. We use v1.0 mainly because previous studies on joint inference reported results w.r.t. v1.0 only.<sup>5</sup> We expect that using the same setup on v2.0 will allow a cross-treebank comparison.<sup>6</sup> We used the first 500 sentences as our dev set and the rest 4500 for training and report our main results on this split. To facilitate the comparison of our results to those reported by (Cohen and Smith, 2007) we use their data set in which 177 empty and “malformed”<sup>7</sup> were removed. The first 3770 trees of the resulting set then were used for training, and the last 418 are used testing. (we ignored the 419 trees in their development set.)

**Morphological Analyzer** Ideally, we would use an of-the-shelf morphological analyzer for mapping each input token to its possible analyses. Such resources exist for Hebrew (Itai et al., 2006), but unfortunately use a tagging scheme which is incom-

<sup>5</sup>The comparison to performance on version 2.0 is meaningless not only because of the change in size, but also conceptual changes in the annotation scheme

<sup>6</sup>Unfortunately running our setup on the v2.0 data set is currently not possible due to missing tokens-morphemes alignment in the v2.0 treebank.

<sup>7</sup>We thank Shay Cohen for providing us with their data set and evaluation Software.

patible with the one of the Hebrew Treebank.<sup>8</sup> For this reason, we use a data-driven morphological analyzer derived from the training data similar to (Cohen and Smith, 2007). We construct a mapping from all the space-delimited tokens seen in the training sentences to their corresponding analyses.

**Lexicon and OOV Handling** Our data-driven morphological-analyzer proposes analyses for unknown tokens as described in Section 5. We use the HSPELL<sup>9</sup> (Har’el and Kenigsberg, 2004) wordlist as a lexeme-based lexicon for pruning segmentations involving invalid segments. Models that employ this strategy are denoted **hsp**. To control for the effect of the HSPELL-based pruning, we also experimented with a morphological analyzer that does not perform this pruning. For these models we limit the options provided for OOV words by not considering the entire token as a valid segmentation in case at least some prefix segmentation exists. This analyzer setting is similar to that of (Cohen and Smith, 2007), and models using it are denoted **nohsp**,

**Parser and Grammar** We used BitPar (Schmid, 2004), an efficient general purpose parser,<sup>10</sup> together with various treebank grammars to parse the input sentences and propose compatible morphological segmentation and syntactic analysis.

We experimented with increasingly rich grammars read off of the treebank. Our first model is **GT<sub>plain</sub>**, a PCFG learned from the treebank after removing all functional features from the syntactic categories. In our second model **GT<sub>vpi</sub>** we also distinguished finite and non-finite verbs and VPs as

<sup>8</sup>Mapping between the two schemes involves non-deterministic many-to-many mappings, and in some cases require a change in the syntactic trees.

<sup>9</sup>An open-source Hebrew spell-checker.

<sup>10</sup>Lattice parsing can be performed by special initialization of the chart in a CKY parser (Chappelier et al., 1999). We currently simulate this by crafting a WCFG and feeding it to BitPar. Given a PCFG grammar  $G$  and a lattice  $L$  with nodes  $n_1 \dots n_k$ , we construct the weighted grammar  $G_L$  as follows: for every arc (lexeme)  $l \in L$  from node  $n_i$  to node  $n_j$ , we add to  $G_L$  the rule  $[l \rightarrow t_{n_i}, t_{n_{i+1}}, \dots, t_{n_j-1}]$  with a probability of 1 (this indicates the lexeme  $l$  spans from node  $n_i$  to node  $n_j$ ).  $G_L$  is then used to parse the string  $t_{n_1} \dots t_{n_{k-1}}$ , where  $t_{n_i}$  is a terminal corresponding to the lattice span between node  $n_i$  and  $n_{i+1}$ . Removing the leaves from the resulting tree yields a parse for  $L$  under  $G$ , with the desired probabilities. We use a patched version of BitPar allowing for direct input of probabilities instead of counts. We thank Felix Hageloh (Hageloh, 2006) for providing us with this version.

proposed in (Tsarfaty, 2006). In our third model  $\mathbf{GT}_{\text{ppp}}$  we also add the distinction between general PPs and possessive PPs following Goldberg and Elhadad (2007). In our fourth model  $\mathbf{GT}_{\text{nph}}$  we add the definiteness status of constituents following Tsarfaty and Sima'an (2007). Finally, model  $\mathbf{GT}_v = 2$  includes parent annotation on top of the various state-splits, as is done also in (Tsarfaty and Sima'an, 2007; Cohen and Smith, 2007). For all grammars, we use fine-grained PoS tags indicating various morphological features annotated therein.

**Evaluation** We use 8 different measures to evaluate the performance of our system on the joint disambiguation task. To evaluate the performance on the segmentation task, we report  $SEG$ , the standard harmonic means for segmentation Precision and Recall  $F_1$  (as defined in Bar-Haim *et al.* (2005); Tsarfaty (2006)) as well as the segmentation accuracy  $SEG_{Tok}$  measure indicating the percentage of input tokens assigned the correct exact segmentation (as reported by Cohen and Smith (2007)).  $SEG_{Tok}(noH)$  is the segmentation accuracy ignoring mistakes involving the implicit definite article  $h$ .<sup>11</sup> To evaluate our performance on the tagging task we report  $CPOS$  and  $FPOS$  corresponding to coarse- and fine-grained PoS tagging results ( $F_1$ ) measure. Evaluating parsing results in our joint framework, as argued by Tsarfaty (2006), is not trivial under the joint disambiguation task, as the hypothesized yield need not coincide with the correct one. Our parsing performance measures ( $SYN$ ) thus report the PARSEVAL extension proposed in Tsarfaty (2006). We further report  $SYN^{CS}$ , the parsing metric of Cohen and Smith (2007), to facilitate the comparison. We report the  $F_1$  value of both measures. Finally, our  $U$  (unparsed) measure is used to report the number of sentences to which our system could not propose a joint analysis.

## 7 Results and Analysis

The accuracy results for segmentation, tagging and parsing using our different models and our standard data split are summarized in Table 1. In addition we report for each model its performance on gold-segmented input ( $GS$ ) to indicate the upper bound

<sup>11</sup>Overt definiteness errors may be seen as a wrong feature rather than as wrong constituent and it is by now an accepted standard to report accuracy with and without such errors.

for the grammars' performance on the parsing task.

The table makes clear that enriching our grammar improves the syntactic performance as well as morphological disambiguation (segmentation and POS tagging) accuracy. This supports our main thesis that decisions taken by single, improved, grammar are beneficial for both tasks. When using the segmentation pruning (using HSPELL) for unseen tokens, performance improves for all tasks as well. Yet we note that the better grammars without pruning outperform the poorer grammars using this technique, indicating that the syntactic context aids, to some extent, the disambiguation of unknown tokens.

Table 2 compares the performance of our system on the setup of Cohen and Smith (2007) to the best results reported by them for the same tasks.

Model	$SEG_{Tok}$	$CPOS$	$FPOS$	$SYN^{CS}$
$\mathbf{GT}_{\text{nohsp/pln}}$	89.50	81.00	77.65	62.22
$\mathbf{GT}_{\text{nohsp}/\dots+\text{nph}}$	89.58	81.26	77.82	64.30
$CS_{\text{pln}}$	91.10	80.40	75.60	64.00
$CS_{v=2}$	90.90	80.50	75.40	64.40
$\mathbf{GT}_{\text{hsp/pln}}$	93.13	83.12	79.12	64.46
$\mathbf{GT}_{\text{nohsp}/\dots+v=2}$	89.66	82.85	78.92	66.31
Oracle $CS_{\text{pln}}$	91.80	83.20	79.10	66.50
Oracle $CS_{v=2}$	91.70	83.00	78.70	67.40
$\mathbf{GT}_{\text{hsp}/\dots+v=2}$	93.38	85.08	80.11	69.11

Table 2: Segmentation, Parsing and Tagging Results using the Setup of (Cohen and Smith, 2007) (sentence length  $\leq 40$ ). The Models' are Ordered by Performance.

We first note that the accuracy results of our system are overall higher on their setup, on all measures, indicating that theirs may be an easier dataset. Secondly, for all our models we provide better fine- and coarse-grained POS-tagging accuracy, and all pruned models outperform the Oracle results reported by them.<sup>12</sup> In terms of syntactic disambiguation, even the simplest grammar pruned with HSPELL outperforms their non-Oracle results. Without HSPELL-pruning, our simpler grammars are somewhat lagging behind, but as the grammars improve the gap is bridged. The addition of vertical markovization enables non-pruned models to outperform all previously reported re-

<sup>12</sup>Cohen and Smith (2007) make use of a parameter ( $\alpha$ ) which is tuned separately for each of the tasks. This essentially means that their model does not result in a true joint inference, as executions for different tasks involve tuning a parameter separately. In our model there are no such hyper-parameters, and the performance is the result of truly joint disambiguation.

Model	U	$SEG_{Tok}$ / no H	$SEG_F$	$CPOS$	$FPOS$	$SYN / SYN^{CS}$	$GS SYN$
<b>GT</b> <sub>nohsp/pln</sub>	7	89.77 / 93.18	91.80	80.36	76.77	60.41 / 61.66	65.00
...+vpi	7	89.80 / 93.18	91.84	80.37	76.74	61.16 / 62.41	66.70
...+ppp	7	89.79 / 93.20	91.86	80.43	76.79	61.47 / 62.86	67.22
...+nph	7	89.78 / 93.20	91.86	80.43	76.87	61.85 / 63.06	68.23
...+v=2	9	89.12 / 92.45	91.77	82.02	77.86	64.53 / 66.02	70.82
<b>GT</b> <sub>hsp/pln</sub>	11	92.00 / 94.81	94.52	82.35	78.11	62.10 / 64.17	65.00
...+vpi	11	92.03 / 94.82	94.58	82.39	78.23	63.00 / 65.06	66.70
...+ppp	11	92.02 / 94.85	94.58	82.48	78.33	63.26 / 65.42	67.22
...+nph	11	92.14 / 94.91	94.73	82.58	78.47	63.98 / 65.98	68.23
...+v=2	13	91.42 / 94.10	94.67	84.23	79.25	66.60 / 68.79	70.82

Table 1: Segmentation, tagging and parsing results on the Standard dev/train Split, for all Sentences

sults. Furthermore, the combination of pruning and vertical markovization of the grammar outperforms the Oracle results reported by Cohen and Smith. This essentially means that a better grammar tunes the joint model for optimized syntactic disambiguation at least in as much as their hyper parameters do. An interesting observation is that while vertical markovization benefits all our models, its effect is less evident in Cohen and Smith.

On the surface, our model may seem as a special case of Cohen and Smith in which  $\alpha = 0$ . However, there is a crucial difference: the morphological probabilities in their model come from discriminative models based on linear context. Many morphological decisions are based on long distance dependencies, and when the global syntactic evidence disagrees with evidence based on local linear context, the two models compete with one another, despite the fact that the PCFG takes also local context into account. In addition, as the CRF and PCFG look at similar sorts of information from within two inherently different models, they are far from independent and optimizing their product is meaningless. Cohen and Smith approach this by introducing the  $\alpha$  hyperparameter, which performs best when optimized independently for each sentence (cf. Oracle results).

In contrast, our morphological probabilities are based on a *unigram*, *lexeme*-based model, and all other (local and non-local) contextual considerations are delegated to the PCFG. This fully generative model caters for real interaction between the syntactic and morphological levels as a part of a single coherent process.

## 8 Discussion and Conclusion

Employing a PCFG-based generative framework to make both syntactic and morphological disambiguation decisions is not only theoretically clean and

linguistically justified and but also probabilistically appropriate and empirically sound. The overall performance of our joint framework demonstrates that a probability distribution obtained over mere syntactic contexts using a Treebank grammar and a data-driven lexicon outperforms upper bounds proposed by previous joint disambiguation systems and achieves segmentation and parsing results on a par with state-of-the-art standalone applications results.

Better grammars are shown here to improve performance on both morphological and syntactic tasks, providing support for the advantage of a joint framework over pipelined or factorized ones. We conjecture that this trend may continue by incorporating additional information, e.g., three-dimensional models as proposed by Tsarfaty and Sima'an (2007). In the current work morphological analyses and lexical probabilities are derived from a small Treebank, which is by no means the best way to go. Using a wide-coverage morphological analyzer based on (Itai et al., 2006) should cater for a better coverage, and incorporating lexical probabilities learned from a big (unannotated) corpus (cf. (Levinger et al., 1995; Goldberg et al., ; Adler et al., 2008)) will make the parser more robust and suitable for use in more realistic scenarios.

**Acknowledgments** We thank Meni Adler and Michael Elhadad (BGU) for helpful comments and discussion. We further thank Khalil Simaan (ILLC-UvA) for his careful advise concerning the formal details of the proposal. The work of the first author was supported by the Lynn and William Frankel Center for Computer Sciences. The work of the second author as well as collaboration visits to Israel was financed by NWO, grant number 017.001.271.

## References

- Meni Adler and Michael Elhadad. 2006. An Unsupervised Morpheme-Based HMM for Hebrew Morphological Disambiguation. In *Proceeding of COLING-ACL-06*, Sydney, Australia.
- Meni Adler, Yoav Goldberg, David Gabay, and Michael Elhadad. 2008. Unsupervised Lexicon-Based Resolution of Unknown Words for Full Morphological Analysis. In *Proceedings of ACL-08*.
- Meni Adler. 2001. Hidden Markov Model for Hebrew Part-of-Speech Tagging. Master's thesis, Ben-Gurion University of the Negev.
- Meni Adler. 2007. *Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach*. Ph.D. thesis, Ben-Gurion University of the Negev, Beer-Sheva, Israel.
- Roy Bar-Haim, Khalil Sima'an, and Yoav Winter. 2005. Choosing an optimal architecture for segmentation and pos-tagging of modern Hebrew. In *Proceedings of ACL-05 Workshop on Computational Approaches to Semitic Languages*.
- Roy Bar-Haim, Khalil Sima'an, and Yoav Winter. 2007. Part-of-speech tagging of Modern Hebrew text. *Natural Language Engineering*, 14(02):223–251.
- J. Chappelier, M. Rajman, R. Aragues, and A. Rozenknop. 1999. Lattice Parsing for Speech Recognition.
- Eugene Charniak, Glenn Carroll, John Adcock, Anthony R. Cassandra, Yoshihiko Gotoh, Jeremy Katz, Michael L. Littman, and John McCann. 1996. Taggers for Parsers. *AI*, 85(1-2):45–57.
- David Chiang, Mona Diab, Nizar Habash, Owen Rambow, and Safiullah Shareef. 2006. Parsing Arabic Dialects. In *Proceedings of EACL-06*.
- Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP-CoNLL-07*, pages 208–217.
- Lewis Glinert. 1989. *The Grammar of Modern Hebrew*. Cambridge University Press.
- Yoav Goldberg and Michael Elhadad. 2007. SVM Model Tampering and Anchored Learning: A Case Study in Hebrew NP Chunking. In *Proceeding of ACL-07*, Prague, Czech Republic.
- Yoav Goldberg, Meni Adler, and Michael Elhadad. EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start), booktitle = Proceedings of ACL-08, year = 2008,.
- Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceeding of ACL-05*.
- Felix Hageloh. 2006. Parsing Using Transforms over Treebanks. Master's thesis, University of Amsterdam.
- Nadav Har'el and Dan Kenigsberg. 2004. HSpell - the free Hebrew Spell Checker and Morphological Analyzer. *Israeli Seminar on Computational Linguistics*.
- Alon Itai, Shuly Wintner, and Shlomo Yona. 2006. A Computational Lexicon of Contemporary Hebrew. In *Proceedings of LREC-06*.
- Moshe Lévinger, Uzi Ornan, and Alon Itai. 1995. Learning Morpholexical Probabilities from an Untagged Corpus with an Application to Hebrew. *Computational Linguistics*, 21:383–404.
- Helmut Schmid, 2000. *LoPar: Design and Implementation*. Institute for Computational Linguistics, University of Stuttgart.
- Helmut Schmid. 2004. Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vector. In *Proceedings of COLING-04*.
- Erel Segal. 2000. Hebrew Morphological Analyzer for Hebrew Undotted Texts. Master's thesis, Technion, Haifa, Israel.
- Danny Shacham and Shuly Wintner. 2007. Morphological Disambiguation of Hebrew: A Case Study in Classifier Combination. In *Proceedings of EMNLP-CoNLL-07*, pages 439–447.
- Khalil Sima'an, Alon Itai, Yoav Winter, Alon Altman, and Noa Nativ. 2001. Building a Tree-Bank for Modern Hebrew Text. In *Traitement Automatique des Langues*, volume 42.
- Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of HLT-05*, pages 475–482, Morristown, NJ, USA. Association for Computational Linguistics.
- Reut Tsarfaty and Yoav Goldberg. 2008. Word-Based or Morpheme-Based? Annotation Strategies for Modern Hebrew Clitics. In *Proceedings of LREC-08*.
- Reut Tsarfaty and Khalil Sima'an. 2004. An Integrated Model for Morphological and Syntactic Disambiguation in Modern Hebrew. MOZAIK detailed proposal, NWO Mozaiek scheme.
- Reut Tsarfaty and Khalil Sima'an. 2007. Three-Dimensional Parametrization for Parsing Morphologically Rich Languages. In *Proceedings of IWPT-07*.
- Reut Tsarfaty. 2006. Integrated Morphological and Syntactic Disambiguation for Modern Hebrew. In *Proceedings of ACL-SRW-06*.
- Shlomo Yona and Shuly Wintner. 2005. A Finite-state Morphological Grammar of Hebrew. In *Proceedings of the ACL-05 Workshop on Computational Approaches to Semitic Languages*.

# Which words are hard to recognize?

## Prosodic, lexical, and disfluency factors that increase ASR error rates

Sharon Goldwater, Dan Jurafsky and Christopher D. Manning

Department of Linguistics and Computer Science

Stanford University

{sgwater, jurafsky, manning}@stanford.edu

### Abstract

Many factors are thought to increase the chances of misrecognizing a word in ASR, including low frequency, nearby disfluencies, short duration, and being at the start of a turn. However, few of these factors have been formally examined. This paper analyzes a variety of lexical, prosodic, and disfluency factors to determine which are likely to increase ASR error rates. Findings include the following. (1) For disfluencies, effects depend on the type of disfluency: errors *increase* by up to 15% (absolute) for words near fragments, but *decrease* by up to 7.2% (absolute) for words near repetitions. This decrease seems to be due to longer word duration. (2) For prosodic features, there are more errors for words with *extreme* values than words with *typical* values. (3) Although our results are based on output from a system with speaker adaptation, speaker differences are a major factor influencing error rates, and the effects of features such as frequency, pitch, and intensity may vary between speakers.

### 1 Introduction

In order to improve the performance of automatic speech recognition (ASR) systems on conversational speech, it is important to understand the factors that cause problems in recognizing words. Previous work on recognition of spontaneous monologues and dialogues has shown that infrequent words are more likely to be misrecognized (Fosler-Lussier and Morgan, 1999; Shinozaki and Furui, 2001) and that fast speech increases error rates (Siegler and Stern, 1995; Fosler-Lussier and Morgan, 1999; Shinozaki

and Furui, 2001). Siegler and Stern (1995) and Shinozaki and Furui (2001) also found higher error rates in very slow speech. Word length (in phones) has also been found to be a useful predictor of higher error rates (Shinozaki and Furui, 2001). In Hirschberg et al.'s (2004) analysis of two human-computer dialogue systems, misrecognized turns were found to have (on average) higher maximum pitch and energy than correctly recognized turns. Results for speech rate were ambiguous: faster utterances had higher error rates in one corpus, but lower error rates in the other. Finally, Adda-Decker and Lamel (2005) demonstrated that both French and English ASR systems had more trouble with male speakers than female speakers, and found several possible explanations, including higher rates of disfluencies and more reduction.

Many questions are left unanswered by these previous studies. In the word-level analyses of Fosler-Lussier and Morgan (1999) and Shinozaki and Furui (2001), only substitution and deletion errors were considered, so we do not know how including insertions might affect the results. Moreover, these studies primarily analyzed lexical, rather than prosodic, factors. Hirschberg et al.'s (2004) work suggests that prosodic factors can impact error rates, but leaves open the question of which factors are important at the word level and how they influence recognition of natural conversational speech. Adda-Decker and Lamel's (2005) suggestion that higher rates of disfluency are a cause of worse recognition for male speakers presupposes that disfluencies raise error rates. While this assumption seems natural, it has yet to be carefully tested, and in particular we do not



know whether disfluent words are associated with errors in adjacent words, or are simply more likely to be misrecognized themselves. Other factors that are often thought to affect a word’s recognition, such as its status as a content or function word, and whether it starts a turn, also remain unexamined.

The present study is designed to address all of these questions by analyzing the effects of a wide range of lexical and prosodic factors on the accuracy of an English ASR system for conversational telephone speech. In the remainder of this paper, we first describe the data set used in our study and introduce a new measure of error, *individual word error rate* (IWER), that allows us to include insertion errors in our analysis, along with deletions and substitutions. Next, we present the features we collected for each word and the effects of those features individually on IWER. Finally, we develop a joint statistical model to examine the effects of each feature while controlling for possible correlations.

## 2 Data

For our analysis, we used the output from the SRI/ICSI/UW RT-04 CTS system (Stolcke et al., 2006) on the NIST RT-03 development set. This system’s performance was state-of-the-art at the time of the 2004 evaluation. The data set contains 36 telephone conversations (72 speakers, 38477 reference words), half from the Fisher corpus and half from the Switchboard corpus.<sup>1</sup>

The standard measure of error used in ASR is *word error rate* (WER), computed as  $100(I + D + S)/R$ , where  $I$ ,  $D$  and  $S$  are the number of insertions, deletions, and substitutions found by aligning the ASR hypotheses with the reference transcriptions, and  $R$  is the number of reference words. Since we wish to know what features of a reference word increase the probability of an error, we need a way to measure the errors attributable to individual words — an *individual word error rate* (IWER). We assume that a substitution or deletion error can be assigned to its corresponding reference word, but for insertion errors, there may be two adjacent reference words that could be responsible. Our solution is to assign any insertion errors to each of

<sup>1</sup>These conversations are not part of the standard Fisher and Switchboard corpora used to train most ASR systems.

	Ins	Del	Sub	Total	% data
Full word	1.6	6.9	10.5	19.0	94.2
Filled pause	0.6	–	16.4	17.0	2.8
Fragment	2.3	–	17.3	19.6	2.0
Backchannel	0.3	30.7	5.0	36.0	0.6
Guess	1.6	–	30.6	32.1	0.4
Total	1.6	6.7	10.9	19.7	100

Table 1: Individual word error rates for different word types, and the proportion of words belonging to each type. Deletions of filled pauses, fragments, and guesses are not counted as errors in the standard scoring method.

the adjacent words. We could then define IWER as  $100(n_i + n_d + n_s)/R$ , where  $n_i$ ,  $n_d$ , and  $n_s$  are the insertion, deletion, and substitution counts for individual words (with  $n_d = D$  and  $n_s = S$ ). In general, however,  $n_i > I$ , so that the IWER for a given data set would be larger than the WER. To facilitate comparisons with standard WER, we therefore discount insertions by a factor  $\alpha$ , such that  $\alpha n_i = I$ . In this study,  $\alpha = .617$ .

## 3 Analysis of individual features

### 3.1 Features

The reference transcriptions used in our analysis distinguish between five different types of words: filled pauses (*um*, *uh*), fragments (*wh-*, *redistr-*), backchannels (*uh-huh*, *mm-hm*), guesses (where the transcribers were unsure of the correct words), and full words (everything else). Error rates for each of these types can be found in Table 1. The remainder of our analysis considers only the 36159 invocabularly full words in the reference transcriptions (70 OOV full words are excluded). We collected the following features for these words:

**Speaker sex** Male or female.

**Broad syntactic class** Open class (e.g., nouns and verbs), closed class (e.g., prepositions and articles), or discourse marker (e.g., *okay*, *well*). Classes were identified using a POS tagger (Ratnaparkhi, 1996) trained on the tagged Switchboard corpus.

**Log probability** The unigram log probability of each word, as listed in the system’s language model.

**Word length** The length of each word (in phones), determined using the most frequent pronunciation

BefRep	FirRep	MidRep	LastRep	AfRep	BefFP	AfFP	BefFr	AfFr
yeah	i	i	i	think	you	should	um	ask for the ref- recommendation

Figure 1: Example illustrating disfluency features: words occurring before and after repetitions, filled pauses, and fragments; first, middle, and last words in a repeated sequence.

found for that word in the recognition lattices.

**Position near disfluency** A collection of features indicating whether a word occurred before or after a filled pause, fragment, or repeated word; or whether the word itself was the first, last, or other word in a sequence of repetitions. Figure 1 illustrates. Only identical repeated words with no intervening words or filled pauses were considered repetitions.

**First word of turn** Turn boundaries were assigned automatically at the beginning of any utterance following a pause of at least 100 ms during which the other speaker spoke.

**Speech rate** The average speech rate (in phones per second) was computed for each utterance using the pronunciation dictionary extracted from the lattices and the utterance boundary timestamps in the reference transcriptions.

In addition to the above features, we used Praat (Boersma and Weenink, 2007) to collect the following additional prosodic features on a subset of the data obtained by excluding all contractions:<sup>2</sup>

**Pitch** The minimum, maximum, mean, and range of pitch for each word.

**Intensity** The minimum, maximum, mean, and range of intensity for each word.

**Duration** The duration of each word.

31017 words (85.8% of the full-word data set) remain in the no-contractions data set after removing words for which pitch and/or intensity features could not be extracted.

<sup>2</sup>Contractions were excluded before collecting prosodic features for the following reason. In the reference transcriptions and alignments used for scoring ASR systems, contractions are treated as two separate words. However, aside from speech rate, our prosodic features were collected using word-by-word timestamps from a forced alignment that used a transcription where contractions are treated as single words. Thus, the start and end times for a contraction in the forced alignment correspond to two words in the alignments used for scoring, and it is not clear how to assign prosodic features appropriately to those words.

## 3.2 Results and discussion

Results of our analysis of individual features can be found in Table 2 (for categorical features) and Figure 2 (for numeric features). Comparing the error rates for the full-word and the no-contractions data sets in Table 2 verifies that removing contractions does not create systematic changes in the patterns of errors, although it does lower error rates (and significance values) slightly overall. (First and middle repetitions are combined as non-final repetitions in the table, because only 52 words were middle repetitions, and their error rates were similar to initial repetitions.)

### 3.2.1 Disfluency features

Perhaps the most interesting result in Table 2 is that the effects of disfluencies are highly variable depending on the type of disfluency and the position of a word relative to it. Non-final repetitions and words next to fragments have an IWER up to 15% (absolute) *higher* than the average word, while final repetitions and words following repetitions have an IWER up to 7.2% *lower*. Words occurring before repetitions or next to filled pauses do not have significantly different error rates than words not in those positions. Our results for repetitions support Shriberg’s (1995) hypothesis that the final word of a repeated sequence is in fact fluent.

### 3.2.2 Other categorical features

Our results support the common wisdom that open class words have lower error rates than other words (although the effect we find is small), and that words at the start of a turn have higher error rates. Also, like Adda-Decker and Lamel (2005), we find that male speakers have higher error rates than females, though in our data set the difference is more striking (3.6% absolute, compared to their 2.0%).

### 3.2.3 Word probability and word length

Turning to Figure 2, we find (consistent with previous results) that low-probability words have dramatically higher error rates than high-probability

		Filled Pau.		Fragment		Repetition				Syntactic Class			Sex		All	
		Bef	Aft	Bef	Aft	Bef	Aft	NonF	Fin	Clos	Open	Disc	1st	M		F
(a)	IWER	17.6	16.9	<b>33.8</b>	<b>21.6</b>	16.7	<b>13.8</b>	<b>26.0</b>	<b>11.6</b>	<b>19.7</b>	<b>18.0</b>	19.6	<b>21.2</b>	<b>20.6</b>	<b>17.0</b>	18.8
	% wds	1.7	1.7	1.6	1.5	0.7	0.9	1.2	1.1	43.8	50.5	5.8	6.2	52.5	47.5	100
(b)	IWER	17.6	17.2	<b>32.0</b>	<b>21.5</b>	15.8	14.2	<b>25.1</b>	<b>11.6</b>	18.8	<b>17.8</b>	19.0	<b>20.3</b>	<b>20.0</b>	<b>16.4</b>	18.3
	% wds	1.9	1.8	1.6	1.5	0.8	0.8	1.4	1.1	43.9	49.6	6.6	6.4	52.2	47.8	100

Table 2: IWER by feature and percentage of words exhibiting each feature for (a) the full-word data set and (b) the no-contractions data set. Error rates that are significantly different for words with and without a given feature (computed using 10,000 samples in a Monte Carlo permutation test) are in **bold** ( $p < .05$ ) or **bold italics** ( $p < .005$ ). Features shown are whether a word occurs before or after a filled pause, fragment, or repetition; is a non-final or final repetition; is open class, closed class, or a discourse marker; is the first word of a turn; or is spoken by a male or female. *All* is the IWER for the entire data set. (Overall IWER is slightly lower than in Table 1 due to the removal of OOV words.)

words. More surprising is that word length in phones does *not* seem to have a consistent effect on IWER. Further analysis reveals a possible explanation: word length is correlated with duration, but anti-correlated to the same degree with log probability (the Kendall  $\tau$  statistics are .50 and -.49). Figure 2 shows that words with longer duration have lower IWER. Since words with more phones tend to have longer duration, but lower frequency, there is no overall effect of length.

### 3.2.4 Prosodic features

Figure 2 shows that means of pitch and intensity have relatively little effect except at extreme values, where more errors occur. In contrast, pitch and intensity range show clear linear trends, with greater range of pitch or intensity leading to lower IWER.<sup>3</sup> As noted above, decreased duration is associated with increased IWER, and (as in previous work), we find that IWER increases dramatically for fast speech. We also see a tendency towards higher IWER for very slow speech, consistent with Shinozaki and Furui (2001) and Siegler and Stern (1995). The effects of pitch minimum and maximum are not shown for reasons of space, but are similar to pitch mean. Also not shown are intensity minimum (with more errors at higher values) and intensity maximum (with more errors at lower values).

For most of our prosodic features, as well as log probability, extreme values seem to be associated

<sup>3</sup>Our decision to use the log transform of pitch range was originally based on the distribution of pitch range values in the data set. Exploratory data analysis also indicated that using the transformed values would likely lead to a better model fit (Section 4) than using the raw values.

with worse recognition than average values. We explore this possibility further in Section 4.

## 4 Analysis using a joint model

In the previous section, we investigated the effects of various individual features on ASR error rates. However, there are many correlations between these features – for example, words with longer duration are likely to have a larger range of pitch and intensity. In this section, we build a single model with all of our features as potential predictors in order to determine the effects of each feature after controlling for the others. We use the no-contractions data set so that we can include prosodic features in our model. Since only 1% of tokens have an IWER  $> 1$ , we simplify modeling by predicting only whether each token is responsible for an error or not. That is, our dependent variable is binary, taking on the value 1 if IWER  $> 0$  for a given token and 0 otherwise.

### 4.1 Model

To model data with a binary dependent variable, a logistic regression model is an appropriate choice. In logistic regression, we model the *log odds* as a linear combination of feature values  $x_0 \dots x_n$ :

$$\log \frac{p}{1-p} = \beta_0 x_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

where  $p$  is the probability that the outcome occurs (here, that a word is misrecognized) and  $\beta_0 \dots \beta_n$  are coefficients (feature weights) to be estimated. Standard logistic regression models assume that all categorical features are *fixed effects*, meaning that all possible values for these features are known in advance, and each value may have an arbitrarily different effect on the outcome. However, features

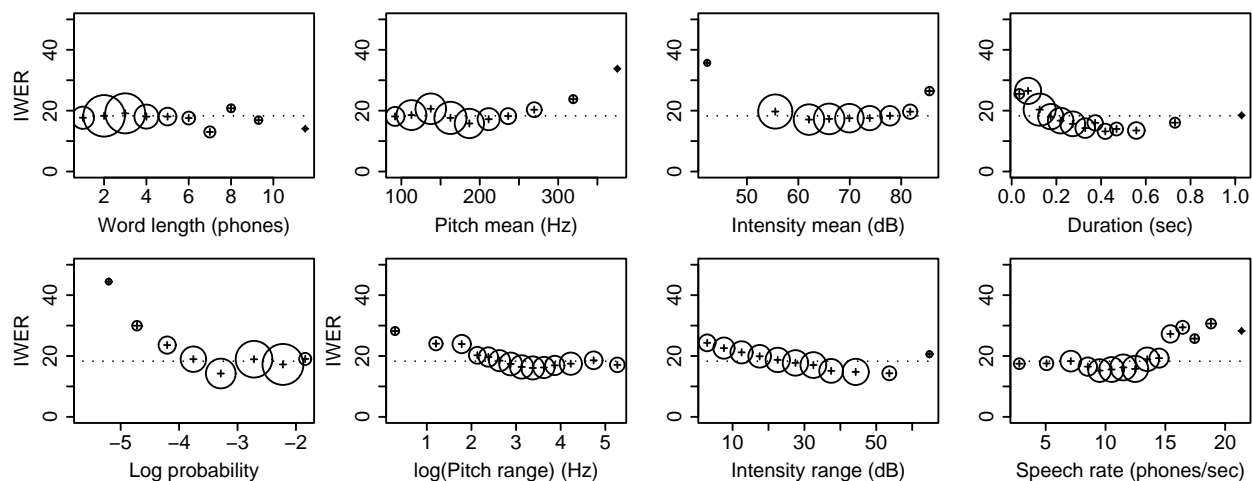


Figure 2: Effects of numeric features on IWER of the SRI system for the no-contractors data set. All feature values were binned, and the average IWER for each bin is plotted, with the area of the surrounding circle proportional to the number of points in the bin. Dotted lines show the average IWER over the entire data set.

such as speaker identity do not fit this pattern. Instead, we control for speaker differences by assuming that speaker identity is a *random effect*, meaning that the speakers observed in the data are a random sample from a larger population. The baseline probability of error for each speaker is therefore assumed to be a normally distributed random variable, with mean equal to the population mean, and variance to be estimated by the model. Stated differently, a random effect allows us to add a factor to the model for speaker identity, without allowing arbitrary variation in error rates between speakers. Models such as ours, with both fixed and random effects, are known as *mixed-effects models*, and are becoming a standard method for analyzing linguistic data (Baayen, 2008). We fit our models using the lme4 package (Bates, 2007) of R (R Development Core Team, 2007).

To analyze the joint effects of all of our features, we initially built as large a model as possible, and used *backwards elimination* to remove features one at a time whose presence did not contribute significantly (at  $p \leq .05$ ) to model fit. All of the features shown in Table 2 were converted to binary variables and included as predictors in our initial model, along with a binary feature controlling for corpus (Fisher or Switchboard), and all numeric features in Figure 2. We did not include minimum and maximum values for pitch and intensity because they are highly

correlated with the mean values, making parameter estimation in the combined model difficult. Preliminary investigation indicated that using the mean values would lead to the best overall fit to the data.

In addition to these basic fixed effects, our initial model included quadratic terms for all of the numeric features, as suggested by our analysis in Section 3, as well as random effects for speaker identity and word identity. All numeric features were rescaled to values between 0 and 1 so that coefficients are comparable.

## 4.2 Results and discussion

Figure 3 shows the estimated coefficients and standard errors for each of the fixed effect categorical features remaining in the reduced model (i.e., after backwards elimination). Since all of the features are binary, a coefficient of  $\beta$  indicates that the corresponding feature, when present, adds a weight of  $\beta$  to the log odds (i.e., multiplies the odds of an error by a factor of  $e^\beta$ ). Thus, features with positive coefficients *increase* the odds of an error, and features with negative coefficients *decrease* the odds of an error. The magnitude of the coefficient corresponds to the size of the effect.

Interpreting the coefficients for our numeric features is less intuitive, since most of these variables have both linear and quadratic effects. The contribution to the log odds of a particular numeric feature

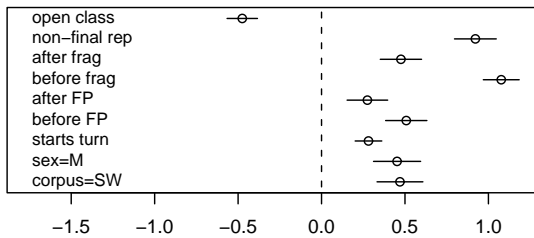


Figure 3: Estimates and standard errors of the coefficients for the categorical predictors in the reduced model.

$x_i$ , with linear and quadratic coefficients  $a$  and  $b$ , is  $ax_i + bx_i^2$ . We plot these curves for each numeric feature in Figure 4. Values on the  $x$  axes with positive  $y$  values indicate increased odds of an error, and negative  $y$  values indicate decreased odds of an error. The  $x$  axes in these plots reflect the rescaled values of each feature, so that 0 corresponds to the minimum value in the data set, and 1 to the maximum value.

#### 4.2.1 Disfluencies

In our analysis of individual features, we found that different types of disfluencies have different effects: non-final repeated words and words near fragments have higher error rates, while final repetitions and words following repetitions have lower error rates. After controlling for other factors, a different picture emerges. There is no longer an effect for final repetitions or words after repetitions; all other disfluency features increase the odds of an error by a factor of 1.3 to 2.9. These differences from Section 3 can be explained by noting that words near filled pauses and repetitions have longer durations than other words (Bell et al., 2003). Longer duration lowers IWER, so controlling for duration reveals the negative effect of the nearby disfluencies. Our results are also consistent with Shriberg’s (1995) findings on fluency in repeated words, since final repetitions have no significant effect in our combined model, while non-final repetitions incur a penalty.

#### 4.2.2 Other categorical features

Without controlling for other lexical or prosodic features, we found that a word is more likely to be misrecognized at the beginning of a turn, and less likely to be misrecognized if it is an open class word. According to our joint model, these effects still hold even after controlling for other features.

Similarly, male speakers still have higher error rates than females. This last result sheds some light on the work of Adda-Decker and Lamel (2005), who suggested several factors that could explain males’ higher error rates. In particular, they showed that males have higher rates of disfluency, produce words with slightly shorter durations, and use more alternate (“sloppy”) pronunciations. Our joint model controls for the first two of these factors, suggesting that the third factor or some other explanation must account for the remaining differences between males and females. One possibility is that female speech is more easily recognized because females tend to have expanded vowel spaces (Diehl et al., 1996), a factor that is associated with greater intelligibility (Bradlow et al., 1996) and is characteristic of genres with lower ASR error rates (Nakamura et al., 2008).

#### 4.2.3 Prosodic features

Examining the effects of pitch and intensity individually, we found that increased range for these features is associated with lower IWER, while higher pitch and extremes of intensity are associated with higher IWER. In the joint model, we see the same effect of pitch mean and an even stronger effect for intensity, with the predicted odds of an error dramatically higher for extreme intensity values. Meanwhile, we no longer see a benefit for increased pitch range and intensity; rather, we see small quadratic effects for both features, i.e. words with average ranges of pitch and intensity are recognized more easily than words with extreme values for these features. As with disfluencies, we hypothesize that the linear trends observed in Section 3 are primarily due to effects of duration, since duration is moderately correlated with both log pitch range ( $\tau = .35$ ) and intensity range ( $\tau = .41$ ).

Our final two prosodic features, duration and speech rate, showed strong linear and weak quadratic trends when analyzed individually. According to our model, both duration and speech rate are still important predictors of error after controlling for other features. However, as with the other prosodic features, predictions of the joint model are dominated by quadratic trends, i.e., predicted error rates are lower for average values of duration and speech rate than for extreme values.

Overall, the results from our joint analysis suggest

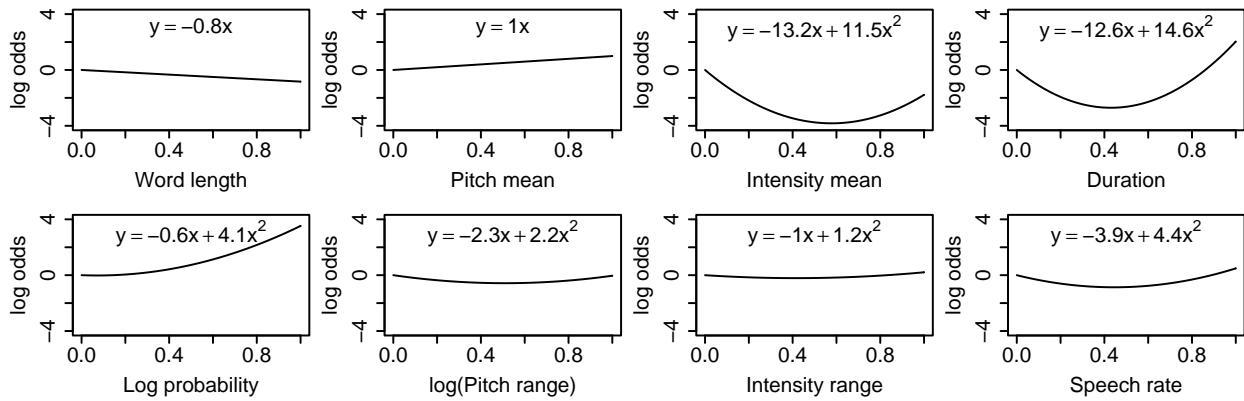


Figure 4: Predicted effect on the log odds of each numeric feature, including linear and (if applicable) quadratic terms.

Model	Neg. log lik.	Diff.	df
Full	12932	0	32
Reduced	12935	3	26
No lexical	13203	271	16
No prosodic	13387	455	20
No speaker	13432	500	31
No word	13267	335	31
Baseline	14691	1759	1

Table 3: Fit to the data of various models. Degrees of freedom (df) for each model is the number of fixed effects plus the number of random effects plus 1 (for the intercept). *Full* model contains all predictors; *Reduced* contains only predictors contributing significantly to fit; *Baseline* contains only intercept. Other models are obtained by removing features from *Full*. *Diff* is the difference in log likelihood between each model and *Full*.

that, after controlling for other factors, *extreme* values for prosodic features are associated with worse recognition than *typical* values.

#### 4.2.4 Differences between lexical items

As discussed above, our model contains a random effect for word identity, to control for the possibility that certain lexical items have higher error rates that are not explained by any of the other factors in the model. It is worth asking whether this random effect is really necessary. To address this question, we compared the fit to the data of two models, each containing all of our fixed effects and a random effect for speaker identity. One model also contained a random effect for word identity. Results are shown in Table 3. The model without a random effect for word identity is significantly worse than the

full model; in fact, this single parameter is more important than all of the lexical features combined. To see which lexical items are causing the most difficulty, we examined the items with the highest estimated increases in error. The top 20 items on this list include *yup*, *yep*, *yes*, *buy*, *then*, *than*, and *r*, all of which are acoustically similar to each other or to other high-frequency words, as well as the words *after*, *since*, *now*, and *though*, which occur in many syntactic contexts, making them difficult to predict based on the language model.

#### 4.2.5 Differences between speakers

We examined the importance of the random effect for speaker identity in a similar fashion to the effect for word identity. As shown in Table 3, speaker identity is a very important factor in determining the probability of error. That is, the lexical and prosodic variables examined here are not sufficient to fully explain the differences in error rates between speakers. In fact, the speaker effect is the single most important factor in the model.

Given that the differences in error rates between speakers are so large (average IWER for different speakers ranges from 5% to 51%), we wondered whether our model is sufficient to capture the kinds of speaker variation that exist. The model assumes that each speaker has a different baseline error rate, but that the effects of each variable are the same for each speaker. Determining the extent to which this assumption is justified is beyond the scope of this paper, however we present some suggestive results in Figure 5. This figure illustrates some of the dif-

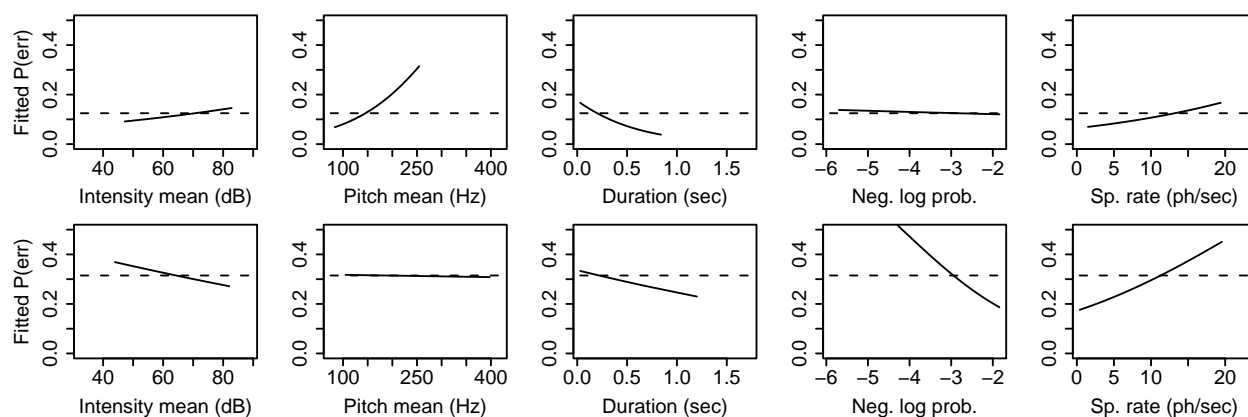


Figure 5: Estimated effects of various features on the error rates of two different speakers (top and bottom). Dashed lines illustrate the baseline probability of error for each speaker. Solid lines were obtained by fitting a logistic regression model to each speaker’s data, with the variable labeled on the  $x$ -axis as the only predictor.

ferences between two speakers chosen fairly arbitrarily from our data set. Not only are the baseline error rates different for the two speakers, but the effects of various features appear to be very different, in one case even reversed. The rest of our data set exhibits similar kinds of variability for many of the features we examined. These differences in ASR behavior between speakers are particularly interesting considering that the system we investigated here already incorporates speaker adaptation models.

## 5 Conclusion

In this paper, we introduced the *individual word error rate* (IWER) for measuring ASR performance on individual words, including insertions as well as deletions and substitutions. Using IWER, we analyzed the effects of various word-level lexical and prosodic features, both individually and in a joint model. Our analysis revealed the following effects. (1) Words at the start of a turn have slightly higher IWER than average, and open class (content) words have slightly lower IWER. These effects persist even after controlling for other lexical and prosodic factors. (2) Disfluencies heavily impact error rates: IWER for non-final repetitions and words adjacent to fragments rises by up to 15% absolute, while IWER for final repetitions and words following repetitions decreases by up to 7.2% absolute. Controlling for prosodic features eliminates the latter benefit, and reveals a negative effect of adjacent filled pauses, suggesting that the effects of these disfluen-

cies are normally obscured by the greater duration of nearby words. (3) For most acoustic-prosodic features, words with extreme values have worse recognition than words with average values. This effect becomes much more pronounced after controlling for other factors. (4) After controlling for lexical and prosodic characteristics, the lexical items with the highest error rates are primarily homophones or near-homophones (e.g., *buy* vs. *by*, *then* vs. *than*). (5) Speaker differences account for much of the variance in error rates between words. Moreover, the direction and strength of effects of different prosodic features may vary between speakers.

While we plan to extend our analysis to other ASR systems in order to determine the generality of our findings, we have already gained important insights into a number of factors that increase ASR error rates. In addition, our results suggest a rich area for future research in further analyzing the variability of both lexical and prosodic effects on ASR behavior for different speakers.

## Acknowledgments

This work was supported by the Edinburgh-Stanford LINK and ONR MURI award N000140510388. We thank Andreas Stolcke for providing the ASR output, language model, and forced alignments used here, and Raghunandan Kumaran and Katrin Kirchhoff for earlier datasets and additional help.

## References

- M. Adda-Decker and L. Lamel. 2005. Do speech recognizers prefer female speakers? In *Proceedings of INTERSPEECH*, pages 2205–2208.
- R. H. Baayen. 2008. *Analyzing Linguistic Data. A Practical Introduction to Statistics*. Cambridge University Press. Prepublication version available at <http://www.mpi.nl/world/persons/private/baayen/publications.html>.
- Douglas Bates, 2007. *lme4: Linear mixed-effects models using S4 classes*. R package version 0.99875-8.
- A. Bell, D. Jurafsky, E. Fosler-Lussier, C. Girand, M. Gregory, and D. Gildea. 2003. Effects of disfluencies, predictability, and utterance position on word form variation in English conversation. *Journal of the Acoustical Society of America*, 113(2):1001–1024.
- P. Boersma and D. Weenink. 2007. Praat: doing phonetics by computer (version 4.5.16). <http://www.praat.org/>.
- A. Bradlow, G. Torretta, and D. Pisoni. 1996. Intelligibility of normal speech I: Global and fine-grained acoustic-phonetic talker characteristics. *Speech Communication*, 20:255–272.
- R. Diehl, B. Lindblom, K. Hoemeke, and R. Fahey. 1996. On explaining certain male-female differences in the phonetic realization of vowel categories. *Journal of Phonetics*, 24:187–208.
- E. Fosler-Lussier and N. Morgan. 1999. Effects of speaking rate and word frequency on pronunciations in conversational speech. *Speech Communication*, 29:137–158.
- J. Hirschberg, D. Litman, and M. Swerts. 2004. Prosodic and other cues to speech recognition failures. *Speech Communication*, 43:155–175.
- M. Nakamura, K. Iwano, and S. Furui. 2008. Differences between acoustic characteristics of spontaneous and read speech and their effects on speech recognition performance. *Computer Speech and Language*, 22:171–184.
- R Development Core Team, 2007. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
- A. Ratnaparkhi. 1996. A Maximum Entropy model for part-of-speech tagging. In *Proceedings of the First Conference on Empirical Methods in Natural Language Processing*, pages 133–142.
- T. Shinozaki and S. Furui. 2001. Error analysis using decision trees in spontaneous presentation speech recognition. In *Proceedings of ASRU 2001*.
- E. Shriberg. 1995. Acoustic properties of disfluent repetitions. In *Proceedings of the International Congress of Phonetic Sciences*, volume 4, pages 384–387.
- M. Siegler and R. Stern. 1995. On the effects of speech rate in large vocabulary speech recognition systems. In *Proceedings of ICASSP*.
- A. Stolcke, B. Chen, H. Franco, V. R. R. Gadde, M. Gra-ciarena, M.-Y. Hwang, K. Kirchhoff, A. Mandal, N. Morgan, X. Lin, T. Ng, M. Ostendorf, K. Sonmez, A. Venkataraman, D. Vergyri, W. Wang, J. Zheng, and Q. Zhu. 2006. Recent innovations in speech-to-text transcription at SRI-ICSI-UW. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5):1729–1744.



# Name Translation in Statistical Machine Translation

## Learning When to Transliterate

**Ulf Hermjakob and Kevin Knight**

University of Southern California  
Information Sciences Institute  
4676 Admiralty Way  
Marina del Rey, CA 90292, USA  
{ulf,knight}@isi.edu

**Hal Daumé III**

University of Utah  
School of Computing  
50 S Central Campus Drive  
Salt Lake City, UT 84112, USA  
me@hal3.name

### Abstract

We present a method to transliterate names in the framework of end-to-end statistical machine translation. The system is trained to learn when to transliterate. For Arabic to English MT, we developed and trained a transliterator on a bitext of 7 million sentences and Google’s English terabyte ngrams and achieved better name translation accuracy than 3 out of 4 professional translators. The paper also includes a discussion of challenges in name translation evaluation.

## 1 Introduction

State-of-the-art statistical machine translation (SMT) is bad at translating names that are not very common, particularly across languages with different character sets and sound systems. For example, consider the following automatic translation:<sup>1</sup>

**Arabic input** موسيقيين مثل باخ وموزار وشوبان  
وبيتهوفن وشومان ورحمانينوف ورافيل  
وبروكوفيف

**SMT output** musicians such as Bach

**Correct translation** composers such as Bach,  
Mozart, Chopin, Beethoven, Schumann,  
Rachmaninoff, Ravel and Prokofiev

The SMT system drops most names in this example. “Name dropping” and mis-translation happens when the system encounters an unknown word, mistakes a name for a common noun, or trains on noisy parallel data. The state-of-the-art is poor for

two reasons. First, although names are important to human readers, automatic MT scoring metrics (such as BLEU) do not encourage researchers to improve name translation in the context of MT. Names are vastly outnumbered by prepositions, articles, adjectives, common nouns, etc. Second, name translation is a hard problem — even professional human translators have trouble with names. Here are four reference translations taken from the same corpus, with mistakes underlined:

**Ref1** composers such as Bach, missing name Chopin, Beethoven, Shumann, Rakmaninov, Ravel and Prokoviev

**Ref2** musicians such as Bach, Mozart, Chopin, Bethoven, Shuman, Rachmaninoff, Rafael and Brokoviev

**Ref3** composers including Bach, Mozart, Schopen, Beethoven, missing name Raphael, Rahmaniev and Brokofien

**Ref4** composers such as Bach, Mozart, missing name Beethoven, Schumann, Rachmaninov, Raphael and Prokofiev

The task of transliterating names (independent of end-to-end MT) has received a significant amount of research, e.g., (Knight and Graehl, 1997; Chen et al., 1998; Al-Onaizan, 2002). One approach is to “sound out” words and create new, plausible target-language spellings that preserve the sounds of the source-language name as much as possible. Another approach is to phonetically match source-language names against a large list of target-language words

<sup>1</sup>taken from NIST02-05 corpora

and phrases. Most of this work has been disconnected from end-to-end MT, a problem which we address head-on in this paper.

The simplest way to integrate name handling into SMT is: (1) run a named-entity identification system on the source sentence, (2) transliterate identified entities with a special-purpose transliteration component, and (3) run the SMT system on the source sentence, as usual, but when looking up phrasal translations for the words identified in step 1, instead use the transliterations from step 2.

Many researchers have attempted this, and it does not work. Typically, translation quality is degraded rather than improved, for the following reasons:

- Automatic named-entity identification makes errors. Some words and phrases that should not be transliterated are nonetheless sent to the transliteration component, which returns a bad translation.
- Not all named entities should be transliterated. Many named entities require a mix of transliteration and translation. For example, in the pair جنوب كاليفورنيا /jnuub kalyfurnya/Southern California, the first Arabic word is translated, and the second word is transliterated.
- Transliteration components make errors. The base SMT system may translate a commonly-occurring name just fine, due to the bitext it was trained on, while the transliteration component can easily supply a worse answer.
- Integration hobbles SMT's use of longer phrases. Even if the named-entity identification and transliteration components operate perfectly, adopting their translations means that the SMT system may no longer have access to longer phrases that include the name. For example, our base SMT system translates رئيس الوزراء لي بنغ (as a whole phrase) to "Premier Li Peng", based on its bitext knowledge. However, if we force لي بنغ to translate as a separate phrase to "Li Peng", then the term رئيس الوزراء becomes ambiguous (with translations including "Prime Minister", "Premier", etc.), and we observe incorrect choices being subsequently made.

To spur better work in name handling, an ACE entity-translation pilot evaluation was recently developed (Day, 2007). This evaluation involves a mixture of entity identification and translation concerns—for example, the scoring system asks for coreference determination, which may or may not be of interest for improving machine translation output.

In this paper, we adopt a simpler metric. We ask: *what percentage of source-language named entities are translated correctly?* This is a precision metric. We can readily apply it to any base SMT system, and to human translations as well. Our goal in augmenting a base SMT system is to increase this percentage. A secondary goal is to make sure that our overall translation quality (as measured by BLEU) does not degrade as a result of the name-handling techniques we introduce. We make all our measurements on an Arabic/English newswire translation task.

Our overall technical approach is summarized here, along with references to sections of this paper:

- We build a component for transliterating between Arabic and English (Section 3).
- We automatically learn to tag those words and phrases in Arabic text, which we believe the transliteration component will translate correctly (Section 4).
- We integrate suggested transliterations into the base SMT search space, with their use controlled by a feature function (Section 5).
- We evaluate both the base SMT system and the augmented system in terms of entity translation accuracy and BLEU (Sections 2 and 6).

## 2 Evaluation

In this section we present the evaluation method that we use to measure our system and also discuss challenges in name transliteration evaluation.

### 2.1 NEWA Evaluation Metric

General MT metrics such as BLEU, TER, METEOR are not suitable for evaluating named entity translation and transliteration, because they are not focused on named entities (NEs). Dropping a comma or a *the* is penalized as much as dropping a name. We therefore use another metric, jointly developed with BBN and LanguageWeaver.

The general idea of the Named Entity Weak Accuracy (NEWA) metric is to

- Count number of NEs in source text: N
- Count number of correctly translated NEs: C
- Divide C/N to get an accuracy figure

In NEWA, an NE is counted as correctly translated if the target reference NE is found in the MT output. The metric has the advantage that it is easy to compute, has no special requirements on an MT system (such as depending on source-target word alignment) and is tokenization independent.

In the result section of this paper, we will use the NEWA metric to measure and compare the accuracy of NE translations in our end-to-end SMT translations and four human reference translations.

## 2.2 Annotated Corpus

BBN kindly provided us with an annotated Arabic text corpus, in which named entities were marked up with their type (e.g. GPE for Geopolitical Entity) and one or more English translations. Example:

عبد الله <PER alt="Abdullah II | Abdallah II">  
<GPE alt="Termoli"> تيرمولى </GPE> فى  
الشانى </PER>

The BBN annotations exhibit a number of issues. For the English translations of the NEs, BBN annotators looked at human reference translations, which may introduce a bias towards those human translations. Specifically, the BBN annotations are sometimes wrong, because the reference translations were wrong. Consider for example the Arabic phrase مصنع بورتران فى تيرمولي (mSn‘ burtran fY tyrmulY), which means *Powertrain plant in Termoli*. The mapping from *tyrmulY* to *Termoli* is not obvious, and even less the one from *burtran* to *Powertrain*. The human reference translations for this phrase are

1. Portran site in Tremolo
2. Termoli plant (*one name dropped*)
3. Portran in Tirnoli
4. Portran assembly plant, in Tirmoli

The BBN annotators adopted the correct translation *Termoli*, but also the incorrect *Portran*. In

other cases the BBN annotators adopted both a correct (Khatami) and an incorrect translation (Khatimi) when referring to the former Iranian president, which would reward a translation with such an incorrect spelling.

- <PER alt="Khatami|Khatimi"> خاتمي </PER>
- <GPE alt="the American"> الاميركية </GPE>

In other cases, all translations are correct, but additional correct translations are missing, as for “the American” above, for which “the US” is an equally valid alternative in the specific sentence it was annotated in.

All this raises the question of what **is** a correct answer. For most Western names, there is normally only one correct spelling. We follow the same conventions as standard media, paying attention to how an organization or individual spells its own name, e.g. Senator Jon Kyl, not Senator John Kyle. For Arabic names, variation is generally acceptable if there is no one clearly dominant spelling in English, e.g. Gaddafi|Gadhafi|Qaddafi|Qadhafi, as long as a given variant is not radically rarer than the most conventional or popular form.

## 2.3 Re-Annotation

Based on the issues we found with the BBN annotations, we re-annotated a sub-corpus of 637 sentences of the BBN gold standard.

We based this re-annotation on detailed annotation guidelines and sample annotations that had previously been developed in cooperation with LanguageWeaver, building on three iterations of test annotations with three annotators.

We checked each NE in every sentence, using human reference translations, automatic transliterator output, performing substantial Web research for many rare names, and checked Google ngrams and counts for the general Web and news archives to determine whether a variant form met our threshold of occurring at least 20% as often as the most dominant form.

## 3 Transliterator

This section describes how we transliterate Arabic words or phrases. Given a word such as رحمانينوف or a phrase such as موريس رافيل, we want to find the English transliteration for it. This is not just a

romanization like *rHmanyuf* and *murys rafyl* for the examples above, but a properly spelled English name such as *Rachmaninoff* and *Maurice Ravel*. The transliteration result can contain several alternatives, e.g. *Rachmaninoff*|*Rachmaninov*. Unlike various generative approaches (Knight and Graehl, 1997; Stalls and Knight, 1998; Li et al., 2004; Matthews, 2007; Sherif and Kondrak, 2007; Kashani et al., 2007), we do not synthesize an English spelling from scratch, but rather find a translation in very large lists of English words (3.4 million) and phrases (47 million).

We develop a similarity metric for Arabic and English words. Since matching against millions of candidates is computationally prohibitive, we store the English words and phrases in an index, such that given an Arabic word or phrase, we quickly retrieve a much smaller set of likely candidates and apply our similarity metric to that smaller list.

We divide the task of transliteration into two steps: given an Arabic word or phrase to transliterate, we (1) identify a list of English transliteration candidates from indexed lists of English words and phrases with counts (section 3.1) and (2) compute for each English name candidate the cost for the Arabic/English name pair (transliteration scoring model, section 3.2).

We then combine the count information with the transliteration cost according to the formula:

$$\text{score}(e) = \log(\text{count}(e))/20 - \text{translit\_cost}(e,f)$$

### 3.1 Indexing with consonant skeletons

We identify a list of English transliteration candidates through what we call a *consonant skeleton* index. Arabic consonants are divided into 11 classes, represented by letters b,f,g,j,k,l,m,n,r,s,t. In a one-time pre-processing step, all 3,420,339 (unique) English words from our English unigram language model (based on Google’s Web terabyte ngram collection) that might be names or part of names (mostly based on capitalization) are mapped to one or more skeletons, e.g.

Rachmaninoff → rkmnnf, rmnnf, rsmnnf, rtsmnnf  
This yields 10,381,377 skeletons (average of 3.0 per word) for which a reverse index is created (with counts). At run time, an Arabic word to be transliterated is mapped to its skeleton, e.g.

رحمانينوف → rmnnf

This skeleton serves as a key for the previously built reverse index, which then yields the list of English candidates with counts:

rmnnf → Rachmaninov (186,216), Rachmaninoff (179,666), Armenonville (3,445), Rachmaninow (1,636), plus 8 others.

Shorter words tend to produce more candidates, resulting in slower transliteration, but since there are relatively few unique short words, this can be addressed by caching transliteration results.

The same consonant skeleton indexing process is applied to name bigrams (47,700,548 unique with 167,398,054 skeletons) and trigrams (46,543,712 unique with 165,536,451 skeletons).

### 3.2 Transliteration scoring model

The cost of an Arabic/English name pair is computed based on 732 rules that assign a cost to a pair of Arabic and English substrings, allowing for one or more context restrictions.

1. ق::q == ::0
2. وف::ough == ::0
3. ح::ch == :[aou],:0.1
4. ق::k == ,\$, \$::0.1 ; ::0.2
5. ء:: == ,EC::0.1

The first example rule above assigns to the straightforward pair ق/q a cost of 0. The second rule includes 2 letters on the Arabic and 4 on the English side. The third rule restricts application to substring pairs where the English side is preceded by the letters a, o, or u. The fourth rule specifies a cost of 0.1 if the substrings occur at the end of (both) names, 0.2 otherwise. According to the fifth rule, the Arabic letter ء may match an empty string on the English side, if there is an English consonant (EC) in the right context of the English side.

The total cost is computed by always applying the longest applicable rule, without branching, resulting in a linear complexity with respect to word-pair length. Rules may include left and/or right context for both Arabic and English. The match fails if no rule applies or the accumulated cost exceeds a preset limit.

Names may have  $n$  words on the English and  $m$  on the Arabic side. For example, *New York* is one word in Arabic and *Abdullah* is two words in Arabic. The

rules handle spaces (as well as digits, apostrophes and other non-alphabetic material) just like regular alphabetic characters, so that our system can handle cases like where words in English and Arabic names do not match one to one.

The French name *Beaujolais* (بوجولييه/bujulyh) deviates from standard English spelling conventions in several places. The accumulative cost from the rules handling these deviations could become prohibitive, with each cost element penalizing the same underlying offense — being French. We solve this problem by allowing for additional context in the form of *style flags*. The rule for matching *eau/و* specifies, in addition to a cost, an (output) style flag +fr (as in French), which in turn serves as an additional context for the rule that matches *ais/يه* at a much reduced cost. Style flags are also used for some Arabic dialects. Extended characters such as é, ö, and § and spelling idiosyncrasies in names on the English side of the bitext that come from various third languages account for a significant portion of the rule set.

Casting the transliteration model as a scoring problem thus allows for very powerful rules with strong contexts. The current set of rules has been built by hand based on a bitext development corpus; future work might include deriving such rules automatically from a training set of transliterated names.

This transliteration scoring model described in this section is used in two ways: (1) to transliterate names at SMT decoding time, and (2) to identify transliteration pairs in a bitext.

## 4 Learning what to transliterate

As already mentioned in the introduction, named entity (NE) identification followed by MT is a bad idea. We don't want to identify NEs per se anyway — we want to identify things that our transliterator will be good at handling, i.e., things that should be transliterated. This might even include loanwords like *bnk* (*bank*) and *brlman* (*parliament*), but would exclude names such as *National Basketball Association* that are often translated rather transliterated.

Our method follows these steps:

1. Take a bitext.
2. Mark the Arabic words and phrases that have a recognizable transliteration on the English side.

3. Remove the English side of the bitext.
4. Divide the annotated Arabic corpus into a training and test corpus.
5. Train a monolingual Arabic tagger to identify which words and phrases (in running Arabic) are good candidates for transliteration (section 4.2)
6. Apply the tagger to test data and evaluate its accuracy.

### 4.1 Mark-up of bitext

Given a tokenized (but unaligned and mixed-case) bitext, we mark up that bitext with links between Arabic and English words that appear to be transliterations. In the following example, linked words are underlined, with numbers indicating what is linked.

**English** The meeting was attended by Omani (1) Secretary of State for Foreign Affairs Yusif (2) bin (3) Alawi (6) bin (8) Abdallah (10) and Special Advisor to Sultan (12) Qabus (13) for Foreign Affairs Umar (14) bin (17) Abdul Munim (19) al-Zawawi (21).

**Arabic (translit.)** uHDr allqa' uzyr aldule al'manY (1) llsh'uun alkhariye yusf (2) bn (3) 'luY (6) bn (8) 'bd allh (10) ualmstshar alkhaS llslTan (12) qabus (13) ll'laqat alkhariye 'mr (14) bn (17) 'bd almn'm (19) alzuauY (21) .

For each Arabic word, the linking algorithm tries to find a matching word on the English side, using the transliteration scoring model described in section 3. If the matcher reaches the end of an Arabic or English word before reaching the end of the other, it continues to “consume” additional words until a word-boundary observing match is found or the cost threshold exceeded.

When there are several viable linking alternatives, the algorithm considers the cost provided by the transliteration scoring model, as well as context to eliminate inferior alternatives, so that for example the different occurrences of the name particle *bin* in the example above are linked to the proper Arabic words, based on the names next to them. The number of links depends, of course, on the specific corpus, but we typically identify about 3.0 links per sentence.

The algorithm is enhanced by a number of heuristics:

- English match candidates are restricted to capitalized words (with a few exceptions).
- We use a list of about 200 Arabic and English stopwords and stopword pairs.
- We use lists of countries and their adjective forms to bridge cross-POS translations such as *Italy's president* on the English and رئيس الايطالي (*Italian president*) on the Arabic side.
- Arabic prefixes such as لـ/ (“to”) are treated in a special way, because they are translated, not transliterated like the rest of the word. Link (12) above is an example.

In this bitext mark-up process, we achieve 99.5% precision and 95% recall based on a manual visualization-tool based evaluation. Of the 5% recall error, 3% are due to noisy data in the bitext such as typos, incorrect translations, or names missing on one side of the bitext.

#### 4.2 Training of Arabic name tagger

The task of the Arabic name tagger (or more precisely, “transliterate-me” tagger) is to predict whether or not a word in an Arabic text should be transliterated, and if so, whether it includes a prefix. Prefixes such as و/u- (“and”) have to be translated rather than transliterated, so it is important to split off any prefix from a name before transliterating that name. This monolingual tagging task is not trivial, as many Arabic words can be both a name and a non-name. For example, الجزيرة (aljzyre) can mean both *Al-Jazeera* and *the island (or peninsula)*.

Features include the word itself plus two words to the left and right, along with various prefixes, suffixes and other characteristics of all of them, totalling about 250 features.

Some of our features depend on large corpus statistics. For this, we divide the tagged Arabic side of our training corpus into a *stat section* and a *core training section*. From the *stat section* we collect statistics as to how often every word, bigram or trigram occurs, and what distribution of name/non-name patterns these ngrams have. The name distribution bigram

الجزيرة الكورية 3327 00:133 01:3193 11:1  
(aljzyre alkurye/“peninsula Korean”) for example tells us that in 3193 out of 3327 occurrences in the

stat corpus bitext, the first word is marked up as a non-name (“0”) and the second as a name (“1”), which strongly suggests that in such a bigram context, *aljzyre* better be **translated** as *island* or *peninsula*, and not be **transliterated** as *Al-Jazeera*.

We train our system on a corpus of 6 million *stat* sentences, and 500,000 *core* training sentences. We employ a sequential tagger trained using the SEARN algorithm (Daumé III et al., 2006) with aggressive updates ( $\beta = 1$ ). Our base learning algorithm is an averaged perceptron, as implemented in the MEGAM package<sup>2</sup>.

Reference	Precision	Recall	F-meas.
Raw test corpus	87.4%	95.7%	91.4%
Adjusted for GS deficiencies	92.1%	95.9%	94.0%

Table 1: Accuracy of “transliterate-me” tagger

Testing on 10,000 sentences, we achieve precision of 87.4% and a recall of 95.7% with respect to the automatically marked-up Gold Standard as described in section 4.1. A manual error analysis of 500 sentences shows that a large portion are not errors after all, but have been marked as errors because of noise in the bitext and errors in the bitext mark-up. After adjusting for these deficiencies in the gold standard, we achieve precision of 92.1% and recall of 95.9% in the name tagging task.

## 5 Integration with SMT

We use the following method to integrate our transliterator into the overall SMT system:

1. We tag the Arabic source text using the tagger described in the previous section.
2. We apply the transliterator described in section 3 to the tagged items. We limit this transliteration to words that occur up to 50 times in the training corpus for single token names (or up to 100 and 150 times for two and three-word names). We do this because the general SMT mechanism tends to do well on more common names, but does poorly on rare names (and will

<sup>2</sup>Freely available at <http://hal3.name/megam>

always drop names it has never seen in the training bitext).

3. On the fly, we add transliterations to SMT phrase table. Instead of a phrasal probability, the transliterations have a special binary feature set to 1. In a tuning step, the Minimum Error Rate Training component of our SMT system iteratively adjusts the set of rule weights, including the weight associated with the transliteration feature, such that the English translations are optimized with respect to a set of known reference translations according to the BLEU translation metric.
4. At run-time, the transliterations then compete with the translations generated by the general SMT system. This means that the MT system will not always use the transliterator suggestions, depending on the combination of language model, translation model, and other component scores.

### 5.1 Multi-token names

We try to transliterate names as much as possible in context. Consider for example the Arabic name:

يوسف أبو صفية ("yusf abu Sfy")

If transliterated as single words without context, the top results would be Joseph|Josef|Yusuf|Yosef|Youssef, Abu|Abo|Ivo|Apo|Ibo, and Sephia|Sofia|Sophia|Safieh|Safia respectively. However, when transliterating the three words together against our list of 47 million English trigrams (section 3), the transliterator will select the (correct) translation *Yousef Abu Safieh*. Note that *Yousef* was not among the top 5 choices, and that *Safieh* was only choice 4.

Similarly, when transliterating وموزار وشوبان ("and Mozart and Chopin") without context, the top results would be Moser|Mauser|Mozer|Mozart|Mouser and Shuppan|Shopping|Schwaben|Schuppan|Shobana (with *Chopin* way down on place 22). Checking our large English lists for a matching *name, name* pattern, the transliterator identifies the correct translation "*Mozart, Chopin*". Note that the transliteration module provides the overall SMT system with up to 5 alternatives, augmented with a choice of English translations for the Arabic prefixes like the comma and the conjunction *and* in the last example.

## 6 End-to-End results

We applied the NEWA metric (section 2) to both our SMT translations as well as the four human reference translations, using both the original named-entity translation annotation and the re-annotation:

Gold Standard	BBN GS	Re-annotated GS
Human 1	87.0%	85.0%
Human 2	85.3%	86.9%
Human 3	90.4%	91.8%
Human 4	86.5%	88.3%
SMT System	80.4%	89.7%

Table 2: Name translation accuracy with respect to BBN and re-annotated Gold Standard on 1730 named entities in 637 sentences.

Almost all scores went up with re-annotations, because the re-annotations more properly reward correct answers.

Based on the original annotations, all human name translations were much better than our SMT system. However, based on our re-annotation, the results are quite different: our system has a higher NEWA score and better name translations than 3 out of 4 human annotators.

The evaluation results confirm that the original annotation method produced a relative bias towards the human translation its annotations were largely based on, compared to other translations.

Table 3 provides more detailed NEWA results. The addition of the transliteration module improves our overall NEWA score from 87.8% to 89.7%, a relative gain of 16% over base SMT system. For names of persons (PER) and facilities (FAC), our system outperforms all human translators. Humans performed much better on Person Nominals (PER.Nom) such as *Swede, Dutchmen, Americans*. Note that name translation quality varies greatly between human translators, with error rates ranging from 8.2-15.0% (absolute).

To make sure our name transliterator does not degrade the overall translation quality, we evaluated our base SMT system with BLEU, as well as our transliteration-augmented SMT system. Our standard newswire training set consists of 10.5 million words of bitext (English side) and 1491 test sen-

NE Type	Count	Baseline SMT	SMT with Transliteration	Human 1	Human 2	Human 3	Human 4
PER	342	266 (77.8%)	280 (81.9%)	210 (61.4%)	265 (77.5%)	278 (81.3%)	275 (80.4%)
GPE	910	863 (94.8%)	877 (96.4%)	867 (95.3%)	849 (93.3%)	885 (97.3%)	852 (93.6%)
ORG	332	280 (84.3%)	282 (84.9%)	263 (79.2%)	265 (79.8%)	293 (88.3%)	281 (84.6%)
FAC	27	18 (66.7%)	24 (88.9%)	21 (77.8%)	20 (74.1%)	22 (81.5%)	20 (74.1%)
PER.Nom	61	49 (80.3%)	48 (78.7%)	61 (100.0%)	56 (91.8%)	60 (98.4%)	57 (93.4%)
LOC	58	43 (74.1%)	41 (70.7%)	48 (82.8%)	48 (82.8%)	51 (87.9%)	43 (74.1%)
All types	1730	1519 (87.8%)	1552 (89.7%)	1470 (85.0%)	1503 (86.9%)	1589 (91.8%)	1528 (88.3%)

Table 3: Name translation accuracy in end-to-end statistical machine translation (SMT) system for different named entity (NE) types: Person (PER), Geopolitical Entity, which includes countries, provinces and towns (GPE), Organization (ORG), Facility (FAC), Nominal Person, e.g. *Swede* (PER.Nom), other location (LOC).

tences. The BLEU scores for the two systems were 50.70 and 50.96 respectively.

Finally, here are end-to-end machine translation results for three sentences, with and without the transliteration module, along with a human reference translation.

*Old:* Al-Basha leads a broad list of musicians such as Bach.

*New:* Al-Basha leads a broad list of musical acts such as Bach, Mozart, Beethoven, Chopin, Schumann, Rachmaninoff, Ravel and Prokofiev.

*Ref:* Al-Bacha performs a long list of works by composers such as Bach, Chopin, Beethoven, Shumann, Rakmaninov, Ravel and Prokofiev.

*Old:* Earlier Israeli military correspondent turn introduction programme "Entertainment Bui"

*New:* Earlier Israeli military correspondent turn to introduction of the programme "Play Boy"

*Ref:* Former Israeli military correspondent turns host for "Playboy" program

*Old:* The Nikkei president company De Beers said that ...

*New:* The company De Beers chairman Nicky Oppenheimer said that ...

*Ref:* Nicky Oppenheimer, chairman of the De Beers company, stated that ...

## 7 Discussion

We have shown that a state-of-the-art statistical machine translation system can benefit from a dedicated transliteration module to improve the transla-

tion of rare names. Improved named entity translation accuracy as measured by the NEWA metric in general, and a reduction in dropped names in particular is clearly valuable to the human reader of machine translated documents as well as for systems using machine translation for further information processing. At the same time, there has been no negative impact on overall quality as measured by BLEU.

We believe that all components can be further improved, e.g.

- Automatically retune the weights in the transliteration scoring model.
- Improve robustness with respect to typos, incorrect or missing translations, and badly aligned sentences when marking up bitexts.
- Add more features for learning whether or not a word should be transliterated, possibly using source language morphology to better identify non-name words never or rarely seen during training.

Additionally, our transliteration method could be applied to other language pairs.

We find it encouraging that we already outperform some professional translators in name translation accuracy. The potential to exceed human translator performance arises from the patience required to translate names right.

## Acknowledgment

This research was supported under DARPA Contract No. HR0011-06-C-0022.



## References

- Yaser Al-Onaizan and Kevin Knight. 2002. Machine Transliteration of Names in Arabic Text. In *Proceedings of the Association for Computational Linguistics Workshop on Computational Approaches to Semitic Languages*.
- Thorsten Brants, Alex Franz. 2006. Web 1T 5-gram Version 1. Released by Google through the Linguistic Data Consortium, Philadelphia, as LDC2006T13.
- Hsin-Hsi Chen, Sheng-Jie Huang, Yung-Wei Ding, and Shih-Chung Tsai. 1998. Proper Name Translation in Cross-Language Information Retrieval. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*.
- Hal Daumé III, John Langford, and Daniel Marcu. 2006. Search-based Structured Prediction. Submitted to the *Machine Learning Journal*. <http://pub.hal3.name/#daume06searn>
- David Day. 2007. Entity Translation 2007 Pilot Evaluation (ET07). In proceedings of the Workshop on Automatic Content Extraction (ACE). College Park, Maryland.
- Byung-Ju Kang and Key-Sun Choi. 2000. Automatic Transliteration and Back-transliteration by Decision Tree Learning. In *Conference on Language Resources and Evaluation*.
- Mehdi M. Kashani, Fred Popowich, and Fatiha Sadat. 2007. Automatic Transliteration of Proper Nouns from Arabic to English. *The Challenge of Arabic For NLP/MT*, 76-84.
- Alexandre Klementiev and Dan Roth. 2006. Named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*.
- Kevin Knight and Jonathan Graehl. 1997. Machine Transliteration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*.
- Li Haizhou, Zhang Min, and Su Jian. 2004. A Joint Source-Channel Model for Machine Transliteration. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*.
- Wei-Hao Lin and Hsin-Hsi Chen. 2002. Backward Machine Transliteration by Learning Phonetic Similarity. *Sixth Conference on Natural Language Learning*, Taipei, Taiwan, 2002.
- David Matthews. 2007. Machine Transliteration of Proper Names. Master's Thesis. School of Informatics. University of Edinburgh.
- Masaaki Nagata, Teruka Saito, and Kenji Suzuki. 2001. Using the Web as a Bilingual Dictionary. In *Proceedings of the Workshop on Data-driven Methods in Machine Translation*.
- Bruno Pouliquen, Ralf Steinberger, Camelia Ignat, Irina Temnikova, Anna Widiger, Wajdi Zaghoulani, and Jan Zizka. 2006. Multilingual Person Name Recognition and Transliteration. CORELA - COgnition, REpresentation, LAnguage, Poitiers, France. Volume 3/3, number 2, pp. 115-123.
- Tarek Sherif and Grzegorz Kondrak. 2007. Substring-Based Transliteration. In *Proceedings of the 45th Annual Meeting on Association for Computational Linguistics*.
- Richard Sproat, ChengXiang Zhai, and Tao Tao. 2006. Named Entity Transliteration with Comparable Corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting on Association for Computational Linguistics*.
- Bonnie Glover Stalls and Kevin Knight. 1998. Translating Names and Technical Terms in Arabic Text. In *Proceedings of the COLING/ACL Workshop on Computational Approaches to Semitic Languages*.
- Stephen Wan and Cornelia Verspoor. 1998. Automatic English-Chinese Name Transliteration for Development of Multilingual Resources. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*. Montreal, Canada.

# Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure

Mark Johnson

Brown University

Mark\_Johnson@Brown.edu

## Abstract

Adaptor grammars (Johnson et al., 2007b) are a non-parametric Bayesian extension of Probabilistic Context-Free Grammars (PCFGs) which in effect learn the probabilities of entire subtrees. In practice, this means that an adaptor grammar learns the structures useful for generating the training data as well as their probabilities. We present several different adaptor grammars that learn to segment phonemic input into words by modeling different linguistic properties of the input. One of the advantages of a grammar-based framework is that it is easy to combine grammars, and we use this ability to compare models that capture different kinds of linguistic structure. We show that incorporating both unsupervised syllabification and collocation-finding into the adaptor grammar significantly improves unsupervised word-segmentation accuracy over that achieved by adaptor grammars that model only one of these linguistic phenomena.

## 1 Introduction

How humans acquire language is arguably the central issue in the scientific study of language. Human language is richly structured, but it is still hotly debated as to whether this structure can be learnt, or whether it must be innately specified. Computational linguistics can contribute to this debate by identifying which aspects of language can potentially be learnt from the input available to a child. Here we try to identify linguistic properties that convey information useful for learning to segment

streams of phonemes into words. We show that simultaneously learning syllable structure and collocations improves word segmentation accuracy compared to models that learn these independently. This suggests that there might be a synergistic interaction in learning several aspects of linguistic structure simultaneously, as compared to learning each kind of linguistic structure independently.

Because learning collocations and word-initial syllable onset clusters requires the learner to be able to identify word boundaries, it might seem that we face a chicken-and-egg problem here. One of the important properties of the adaptor grammar inference procedure is that it gives us a way of learning these interacting linguistic structures simultaneously.

Adaptor grammars are also interesting because they can be viewed as directly inferring linguistic structure. Most well-known machine-learning and statistical inference procedures are parameter estimation procedures, i.e., the procedure is designed to find the values of a finite vector of parameters. Standard methods for learning linguistic structure typically try to reduce structure learning to parameter estimation, say, by using an iterative generate-and-prune procedure in which each iteration consists of a rule generation step that proposes new rules according to some scheme, a parameter estimation step that estimates the utility of these rules, and pruning step that removes low utility rules. For example, the Bayesian unsupervised PCFG estimation procedure devised by Stolcke (1994) uses a model-merging procedure to propose new sets of PCFG rules and a Bayesian version of the EM procedure to estimate their weights.

Recently, methods have been developed in the statistical community for Bayesian inference of increasingly sophisticated non-parametric models. (“Non-parametric” here means that the models are not characterized by a finite vector of parameters, so the complexity of the model can vary depending on the data it describes). Adaptor grammars are a framework for specifying a wide range of such models for grammatical inference. They can be viewed as a nonparametric extension of PCFGs.

Informally, there seem to be at least two natural ways to construct non-parametric extensions of a PCFG. First, we can construct an infinite number of more specialized PCFGs by splitting or refining the PCFG’s nonterminals into increasingly finer states; this leads to the iPCFG or “infinite PCFG” (Liang et al., 2007). Second, we can generalize over arbitrary subtrees rather than local trees in much the way done in DOP or tree substitution grammar (Bod, 1998; Joshi, 2003), which leads to adaptor grammars.

Informally, the units of generalization of adaptor grammars are entire subtrees, rather than just local trees, as in PCFGs. Just as in tree substitution grammars, each of these subtrees behaves as a new context-free rule that expands the subtree’s root node to its leaves, but unlike a tree substitution grammar, in which the subtrees are specified in advance, in an adaptor grammar the subtrees, as well as their probabilities, are learnt from the training data. In order to make parsing and inference tractable we require the leaves of these subtrees to be terminals, as explained in section 2. Thus adaptor grammars are simple models of structure learning, where the subtrees that constitute the units of generalization are in effect new context-free rules learnt during the inference process. (In fact, the inference procedure for adaptor grammars described in Johnson et al. (2007b) relies on a PCFG approximation that contains a rule for each subtree generalization in the adaptor grammar).

This paper applies adaptor grammars to word segmentation and morphological acquisition. Linguistically, these exhibit considerable cross-linguistic variation, and so are likely to be learned by human learners. It’s also plausible that semantics and contextual information is less important for their acquisition than, say, syntax.

## 2 From PCFGs to Adaptor Grammars

This section introduces adaptor grammars as an extension of PCFGs; for a more detailed exposition see Johnson et al. (2007b). Formally, an adaptor grammar is a PCFG in which a subset  $M$  of the nonterminals are *adapted*. An adaptor grammar generates the same set of trees as the CFG with the same rules, but instead of defining a fixed probability distribution over these trees as a PCFG does, it defines a distribution over distributions over trees. An adaptor grammar can be viewed as a kind of PCFG in which each subtree of each adapted nonterminal  $A \in M$  is a potential rule, with its own probability, so an adaptor grammar is nonparametric if there are infinitely many possible adapted subtrees. (An adaptor grammar can thus be viewed as a tree substitution grammar with infinitely many initial trees). But any finite set of sample parses for any finite corpus can only involve a finite number of such subtrees, so the corresponding PCFG approximation only involves a finite number of rules, which permits us to build MCMC samplers for adaptor grammars.

A PCFG can be viewed as a set of recursively-defined mixture distributions  $G_A$  over trees, one for each nonterminal and terminal in the grammar. If  $A$  is a terminal then  $G_A$  is the distribution that puts all of its mass on the unit tree (i.e., tree consisting of a single node) labeled  $A$ . If  $A$  is a nonterminal then  $G_A$  is the distribution over trees with root labeled  $A$  that satisfies:

$$G_A = \sum_{A \rightarrow B_1 \dots B_n \in R_A} \theta_{A \rightarrow B_1 \dots B_n} \text{TD}_A(G_{B_1}, \dots, G_{B_n})$$

where  $R_A$  is the set of rules expanding  $A$ ,  $\theta_{A \rightarrow B_1, \dots, B_n}$  is the PCFG “probability” parameter associated with the rule  $A \rightarrow B_1 \dots B_n$  and  $\text{TD}_A(G_{B_1}, \dots, G_{B_n})$  is the distribution over trees with root label  $A$  satisfying:

$$\text{TD}_A(G_1, \dots, G_n) \left( \begin{array}{c} A \\ \diagdown \quad \diagup \\ t_1 \quad \dots \quad t_n \end{array} \right) = \prod_{i=1}^n G_i(t_i).$$

That is,  $\text{TD}_A(G_1, \dots, G_n)$  is the distribution over trees whose root node is labeled  $A$  and each subtree  $t_i$  is generated *independently* from the distribution  $G_i$ . This independence assumption is what makes a PCFG “context-free” (i.e., each subtree is independent given its label). Adaptor grammars relax

this independence assumption by in effect learning the probability of the subtrees rooted in a specified subset  $M$  of the nonterminals known as the *adapted nonterminals*.

Adaptor grammars achieve this by associating each adapted nonterminal  $A \in M$  with a Dirichlet Process (DP). A DP is a function of a *base distribution*  $H$  and a *concentration parameter*  $\alpha$ , and it returns a distribution over distributions  $\text{DP}(\alpha, H)$ . There are several different ways to define DPs; one of the most useful is the characterization of the conditional or sampling distribution of a draw from  $\text{DP}(\alpha, H)$  in terms of the Polya urn or Chinese Restaurant Process (Teh et al., 2006). The Polya urn initially contains  $\alpha H(x)$  balls of color  $x$ . We sample a distribution from  $\text{DP}(\alpha, H)$  by repeatedly drawing a ball at random from the urn and then returning it plus an additional ball of the same color to the urn.

In an adaptor grammar there is one DP for each adapted nonterminal  $A \in M$ , whose base distribution  $H_A$  is the distribution over trees defined using  $A$ 's PCFG rules. This DP ‘‘adapts’’  $A$ 's PCFG distribution by moving mass from the infrequently to the frequently occurring subtrees. An adaptor grammar associates a distribution  $G_A$  that satisfies the following constraints with each nonterminal  $A$ :

$$\begin{aligned} G_A &\sim \text{DP}(\alpha_A, H_A) && \text{if } A \in M \\ G_A &= H_A && \text{if } A \notin M \\ H_A &= \sum_{A \rightarrow B_1 \dots B_n, B_i \in R_A} \theta_{A \rightarrow B_1 \dots B_n} \text{TD}_A(G_{B_1}, \dots, G_{B_n}) \end{aligned}$$

Unlike a PCFG, an adaptor grammar does not define a single distribution over trees; rather, each set of draws from the DPs defines a different distribution. In the adaptor grammars used in this paper there is no recursion amongst adapted nonterminals (i.e., an adapted nonterminal never expands to itself); it is currently unknown whether there are tree distributions that satisfy the adaptor grammar constraints for recursive adaptor grammars.

Inference for an adaptor grammar involves finding the rule probabilities  $\theta$  and the adapted distributions over trees  $G$ . We put Dirichlet priors over the rule probabilities, i.e.:

$$\theta_A \sim \text{DIR}(\beta_A)$$

where  $\theta_A$  is the vector of probabilities for the rules

expanding the nonterminal  $A$  and  $\beta_A$  are the corresponding Dirichlet parameters.

The applications described below require unsupervised estimation, i.e., the training data consists of terminal strings alone. Johnson et al. (2007b) describe an MCMC procedure for inferring the adapted tree distributions  $G_A$ , and Johnson et al. (2007a) describe a Bayesian inference procedure for the PCFG rule parameters  $\theta$  using a Metropolis-Hastings MCMC procedure; implementations are available from the author’s web site.

Informally, the inference procedure proceeds as follows. We initialize the sampler by randomly assigning each string in the training corpus a random tree generated by the grammar. Then we randomly select a string to resample, and sample a parse of that string with a PCFG approximation to the adaptor grammar. This PCFG contains a production for each adapted subtree in the parses of the other strings in the training corpus. A final accept-reject step corrects for the difference in the probability of the sampled tree under the adaptor grammar and the PCFG approximation.

### 3 Word segmentation with adaptor grammars

We now turn to linguistic applications of adaptor grammars, specifically, to models of unsupervised word segmentation. We follow previous work in using the Brent corpus consists of 9790 transcribed utterances (33,399 words) of child-directed speech from the Bernstein-Ratner corpus (Bernstein-Ratner, 1987) in the CHILDES database (MacWhinney and Snow, 1985). The utterances have been converted to a phonemic representation using a phonemic dictionary, so that each occurrence of a word has the same phonemic transcription. Utterance boundaries are given in the input to the system; other word boundaries are not. We evaluated the f-score of the recovered word constituents (Goldwater et al., 2006b). Using the adaptor grammar software available on the author’s web site, samplers were run for 10,000 epochs (passes through the training data). We scored the parses assigned to the training data at the end of sampling, and for the last two epochs we annealed at temperature 0.5 (i.e., squared the probability) during sampling in or-

	1	10	100	1000
U word	<b>0.55</b>	<b>0.55</b>	<b>0.55</b>	0.53
U morph	<b>0.46</b>	<b>0.46</b>	0.42	0.36
U syll	<b>0.52</b>	0.51	0.49	0.46
C word	0.53	0.64	0.74	<b>0.76</b>
C morph	0.56	0.63	<b>0.73</b>	0.63
C syll	0.77	0.77	<b>0.78</b>	0.74

Table 1: Word segmentation f-score results for all models, as a function of DP concentration parameter  $\alpha$ . “U” indicates unigram-based grammars, while “C” indicates collocation-based grammars.

Sentence  $\rightarrow$  Word<sup>+</sup>  
Word  $\rightarrow$  Phoneme<sup>+</sup>

Figure 1: The unigram word adaptor grammar, which uses a unigram model to generate a sequence of words, where each word is a sequence of phonemes. Adapted nonterminals are underlined.

der to concentrate mass on high probability parses. In all experiments below we set  $\beta = 1$ , which corresponds to a uniform prior on PCFG rule probabilities  $\theta$ . We tied the Dirichlet Process concentration parameters  $\alpha$ , and performed runs with  $\alpha = 1, 10, 100$  and 1000; apart from this, no attempt was made to optimize the hyperparameters. Table 1 summarizes the word segmentation f-scores for all models described in this paper.

### 3.1 Unigram word adaptor grammar

Johnson et al. (2007a) presented an adaptor grammar that defines a unigram model of word segmentation and showed that it performs as well as the unigram DP word segmentation model presented by (Goldwater et al., 2006a). The adaptor grammar that encodes a unigram word segmentation model shown in Figure 1.

In this grammar and the grammars below, underlining indicates an adapted nonterminal. Phoneme is a nonterminal that expands to each of the 50 distinct phonemes present in the Brent corpus. This grammar defines a Sentence to consist of a sequence of Words, where a Word consists of a sequence of Phonemes. The category Word is adapted, which means that the grammar learns the words that occur in the training corpus. We present our adap-

Sentence  $\rightarrow$  Words  
Words  $\rightarrow$  Word  
Words  $\rightarrow$  Word Words  
Word  $\rightarrow$  Phonemes  
Phonemes  $\rightarrow$  Phoneme  
Phonemes  $\rightarrow$  Phoneme Phonemes

Figure 2: The unigram word adaptor grammar of Figure 1 where regular expressions are expanded using new unadapted right-branching nonterminals.

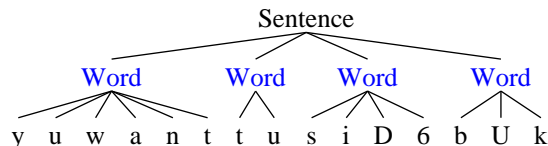


Figure 3: A parse of the phonemic representation of “you want to see the book” produced by unigram word adaptor grammar of Figure 1. Only nonterminal nodes labeled with adapted nonterminals and the start symbol are shown.

tor grammars using regular expressions for clarity, but since our implementation does not handle regular expressions in rules, in the grammars actually used by the program they are expanded using new non-adapted nonterminals that rewrite in a uniform right-branching manner. That is, the adaptor grammar used by the program is shown in Figure 2.

The unigram word adaptor grammar generates parses such as the one shown in Figure 3. With  $\alpha = 1$  and  $\alpha = 10$  we obtained a word segmentation f-score of 0.55. Depending on the run, between 1, 100 and 1, 400 subtrees (i.e., new rules) were found for Word. As reported in Goldwater et al. (2006a) and Goldwater et al. (2007), a unigram word segmentation model tends to undersegment and misanalyse collocations as individual words. This is presumably because the unigram model has no way to capture dependencies between words in collocations except to make the collocation into a single word.

### 3.2 Unigram morphology adaptor grammar

This section investigates whether learning morphology together with word segmentation improves word segmentation accuracy. Johnson et al. (2007a) presented an adaptor grammar for segmenting verbs into stems and suffixes that implements the DP-

Sentence  $\rightarrow$  Word<sup>+</sup>  
 Word  $\rightarrow$  Stem (Suffix)  
 Stem  $\rightarrow$  Phoneme<sup>+</sup>  
 Suffix  $\rightarrow$  Phoneme<sup>+</sup>

Figure 4: The unigram morphology adaptor grammar, which generates each Sentence as a sequence of Words, and each Word as a Stem optionally followed by a Suffix. Parentheses indicate optional constituents.

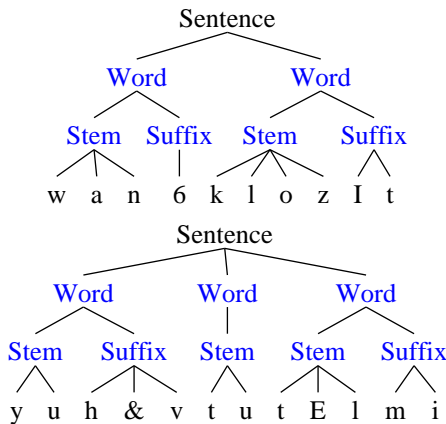


Figure 5: Parses of “wanna close it” and “you have to tell me” produced by the unigram morphology grammar of Figure 4. The first parse was chosen because it demonstrates how the grammar is intended to analyse “wanna” into a Stem and Suffix, while the second parse shows how the grammar tends to use Stem and Suffix to capture collocations.

based unsupervised morphological analysis model presented by Goldwater et al. (2006b). Here we combine that adaptor grammar with the unigram word segmentation grammar to produce the adaptor grammar shown in Figure 4, which is designed to simultaneously learn both word segmentation and morphology.

Parentheses indicate optional constituents in these rules, so this grammar says that a Sentence consists of a sequence of Words, and each Word consists of a Stem followed by an optional Suffix. The categories Word, Stem and Suffix are adapted, which means that the grammar learns the Words, Stems and Suffixes that occur in the training corpus. Technically this grammar implements a *Hierarchical Dirichlet Process* (HDP) (Teh et al., 2006) because the base distribution for the Word DP is itself constructed from the Stem and Suffix distributions, which are

themselves generated by DPs.

This grammar recovers words with an f-score of only 0.46 with  $\alpha = 1$  or  $\alpha = 10$ , which is considerably less accurate than the unigram model of section 3.1. Typical parses are shown in Figure 5. The unigram morphology grammar tends to misanalyse even longer collocations as words than the unigram word grammar does. Inspecting the parses shows that rather than capturing morphological structure, the Stem and Suffix categories typically expand to words themselves, so the Word category expands to a collocation. It may be possible to correct this by “tuning” the grammar’s hyperparameters, but we did not attempt this here.

These results are not too surprising, since the kind of regular stem-suffix morphology that this grammar can capture is not common in the Brent corpus. It is possible that a more sophisticated model of morphology, or even a careful tuning of the Bayesian prior parameters  $\alpha$  and  $\beta$ , would produce better results.

### 3.3 Unigram syllable adaptor grammar

PCFG estimation procedures have been used to model the supervised and unsupervised acquisition of syllable structure (Müller, 2001; Müller, 2002); and the best performance in unsupervised acquisition is obtained using a grammar that encodes linguistically detailed properties of syllables whose rules are inferred using a fairly complex algorithm (Goldwater and Johnson, 2005). While that work studied the acquisition of syllable structure from isolated words, here we investigate whether learning syllable structure together with word segmentation improves word segmentation accuracy. Modeling syllable structure is a natural application of adaptor grammars, since the grammar can learn the possible onset and coda clusters, rather than requiring them to be stipulated in the grammar.

In the unigram syllable adaptor grammar shown in Figure 7, Consonant expands to any consonant and Vowel expands to any vowel. This grammar defines a Word to consist of up to three Syllables, where each Syllable consists of an Onset and a Rhyme and a Rhyme consists of a Nucleus and a Coda. Following Goldwater and Johnson (2005), the grammar differentiates between OnsetI, which expands to word-initial onsets, and Onset,

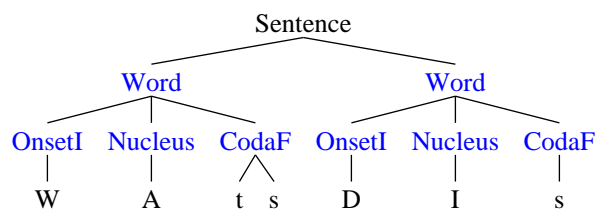


Figure 6: A parse of “what’s this” produced by the unigram syllable adaptor grammar of Figure 7. (Only adapted non-root nonterminals are shown in the parse).

which expands to non-word-initial onsets, and between CodaF, which expands to word-final codas, and Coda, which expands to non-word-final codas. Note that we do not need to distinguish specific positions within the Onset and Coda clusters as Goldwater and Johnson (2005) did, since the adaptor grammar learns these clusters directly. Just like the unigram morphology grammar, the unigram syllable grammar also defines a HDP because the base distribution for Word is defined in terms of the Onset and Rhyme distributions.

The unigram syllable grammar achieves a word segmentation f-score of 0.52 at  $\alpha = 1$ , which is also lower than the unigram word grammar achieves. Inspection of the parses shows that the unigram syllable grammar also tends to misanalyse long collocations as Words. Specifically, it seems to misanalyse function words as associated with the content words next to them, perhaps because function words tend to have simpler initial and final clusters.

We cannot compare our syllabification accuracy with Goldwater’s and others’ previous work because that work used different, supervised training data and phonological representations based on British rather than American pronunciation.

### 3.4 Collocation word adaptor grammar

Goldwater et al. (2006a) showed that modeling dependencies between adjacent words dramatically improves word segmentation accuracy. It is not possible to write an adaptor grammar that directly implements Goldwater’s bigram word segmentation model because an adaptor grammar has one DP per adapted nonterminal (so the number of DPs is fixed in advance) while Goldwater’s bigram model has one DP per word type, and the number of word types is not known in advance. However it is pos-

- Sentence  $\rightarrow$  Word<sup>+</sup>
- Word  $\rightarrow$  SyllableIF
- Word  $\rightarrow$  SyllableI SyllableF
- Word  $\rightarrow$  SyllableI Syllable SyllableF
- Syllable  $\rightarrow$  (Onset) Rhyme
- SyllableI  $\rightarrow$  (OnsetI) Rhyme
- SyllableF  $\rightarrow$  (Onset) RhymeF
- SyllableIF  $\rightarrow$  (OnsetI) RhymeF
- Rhyme  $\rightarrow$  Nucleus (Coda)
- RhymeF  $\rightarrow$  Nucleus (CodaF)
- Onset  $\rightarrow$  Consonant<sup>+</sup>
- OnsetI  $\rightarrow$  Consonant<sup>+</sup>
- Coda  $\rightarrow$  Consonant<sup>+</sup>
- CodaF  $\rightarrow$  Consonant<sup>+</sup>
- Nucleus  $\rightarrow$  Vowel<sup>+</sup>

Figure 7: The unigram syllable adaptor grammar, which generates each word as a sequence of up to three Syllables. Word-initial Onsets and word-final Codas are distinguished using the suffixes “I” and “F” respectively; these are propagated through the grammar to ensure that these appear in the correct positions.

- Sentence  $\rightarrow$  Colloc<sup>+</sup>
- Colloc  $\rightarrow$  Word<sup>+</sup>
- Word  $\rightarrow$  Phoneme<sup>+</sup>

Figure 8: The collocation word adaptor grammar, which generates a Sentence as sequence of Colloc(ations), each of which consists of a sequence of Words.

sible for an adaptor grammar to generate a sentence as a sequence of *collocations*, each of which consists of a sequence of words. These collocations give the grammar a way to model dependencies between words.

With the DP concentration parameters  $\alpha = 1000$  we obtained a f-score of 0.76, which is approximately the same as the results reported by Goldwater et al. (2006a) and Goldwater et al. (2007). This suggests that the collocation word adaptor grammar can capture inter-word dependencies similar to those that improve the performance of Goldwater’s bigram segmentation model.

### 3.5 Collocation morphology adaptor grammar

One of the advantages of working within a grammatical framework is that it is often easy to combine

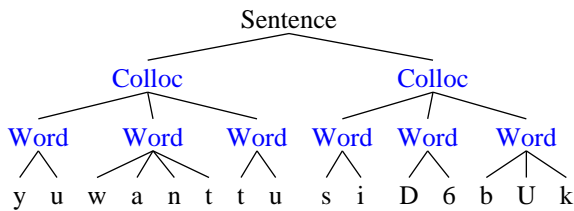


Figure 9: A parse of “you want to see the book” produced by the collocation word adaptor grammar of Figure 8.

$$\begin{aligned}
 \text{Sentence} &\rightarrow \text{Colloc}^+ \\
 \text{Colloc} &\rightarrow \text{Word}^+ \\
 \text{Word} &\rightarrow \text{Stem (Suffix)} \\
 \text{Stem} &\rightarrow \text{Phoneme}^+ \\
 \text{Suffix} &\rightarrow \text{Phoneme}^+
 \end{aligned}$$

Figure 10: The collocation morphology adaptor grammar, which generates each Sentence as a sequence of Colloc(ations), each Colloc as a sequence of Words, and each Word as a Stem optionally followed by a Suffix.

different grammar fragments into a single grammar. In this section we combine the collocation aspect of the previous grammar with the morphology component of the grammar presented in section 3.2 to produce a grammar that generates Sentences as sequences of Colloc(ations), where each Colloc consists of a sequence of Words, and each Word consists of a Stem followed by an optional Suffix, as shown in Figure 10.

This grammar achieves a word segmentation f-score of 0.73 at  $\alpha = 100$ , which is much better than the unigram morphology grammar of section 3.2, but not as good as the collocation word grammar of the previous section. Inspecting the parses shows

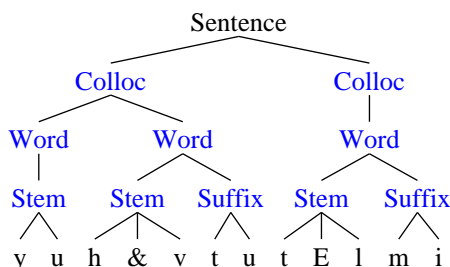


Figure 11: A parse of the phonemic representation of “you have to tell me” using the collocation morphology adaptor grammar of Figure 10.

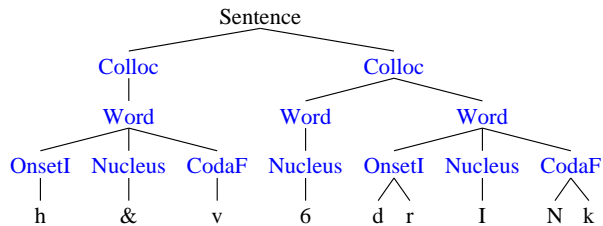


Figure 12: A parse of “have a drink” produced by the collocation syllable adaptor grammar. (Only adapted non-root nonterminals are shown in the parse).

that while the ability to directly model collocations reduces the number of collocations misanalysed as words, function words still tend to be misanalysed as morphemes of two-word collocations. In fact, some of the misanalyses have a certain plausibility to them (e.g., “to” is often analysed as the suffix of verbs such as “have”, “want” and “like”, while “me” is often analysed as a suffix of verbs such as “show” and “tell”), but they lower the word f-score considerably.

### 3.6 Collocation syllable adaptor grammar

The collocation syllable adaptor grammar is the same as the unigram syllable adaptor grammar of Figure 7, except that the first production is replaced with the following pair of productions.

$$\begin{aligned}
 \text{Sentence} &\rightarrow \text{Colloc}^+ \\
 \text{Colloc} &\rightarrow \text{Word}^+
 \end{aligned}$$

This grammar generates a Sentence as a sequence of Colloc(ations), each of which is composed of a sequence of Words, each of which in turn is composed of a sequence of Syll(ables).

This grammar achieves a word segmentation f-score of 0.78 at  $\alpha = 100$ , which is the highest f-score of any of the grammars investigated in this paper, including the collocation word grammar, which models collocations but not syllables. To confirm that the difference is significant, we ran a Wilcoxon test to compare the f-scores obtained from 8 runs of the collocation syllable grammar with  $\alpha = 100$  and the collocation word grammar with  $\alpha = 1000$ , and found that the difference is significant at  $p = 0.006$ .

## 4 Conclusion and future work

This paper has shown how adaptor grammars can be used to study a variety of different linguistic hy-



potheses about the interaction of morphology and syllable structure with word segmentation. Technically, adaptor grammars are a way of specifying a variety of Hierarchical Dirichlet Processes (HDPs) that can spread their support over an unbounded number of distinct subtrees, giving them the ability to learn which subtrees are most useful for describing the training corpus. Thus adaptor grammars move beyond simple parameter estimation and provide a principled approach to the Bayesian estimation of at least some types of linguistic structure. Because of this, less linguistic structure needs to be “built in” to an adaptor grammar compared to a comparable PCFG. For example, the adaptor grammars for syllable structure presented in sections 3.3 and 3.6 learn more information about syllable onsets and codas than the PCFGs presented in Goldwater and Johnson (2005).

We used adaptor grammars to study the effects of modeling morphological structure, syllabification and collocations on the accuracy of a standard unsupervised word segmentation task. We showed how adaptor grammars can implement a previously investigated model of unsupervised word segmentation, the unigram word segmentation model. We then investigated adaptor grammars that incorporate one additional kind of information, and found that modeling collocations provides the greatest improvement in word segmentation accuracy, resulting in a model that seems to capture many of the same interword dependencies as the bigram model of Goldwater et al. (2006b).

We then investigated grammars that combine these kinds of information. There does not seem to be a straight forward way to design an adaptor grammar that models both morphology and syllable structure, as morpheme boundaries typically do not align with syllable boundaries. However, we showed that an adaptor grammar that models collocations and syllable structure performs word segmentation more accurately than an adaptor grammar that models either collocations or syllable structure alone. This is not surprising, since syllable onsets and codas that occur word-peripherally are typically different to those that appear word-internally, and our results suggest that by tracking these onsets and codas, it is possible to learn more accurate word segmentation.

There are a number of interesting directions for future work. In this paper all of the hyperparameters  $\alpha_A$  were tied and varied simultaneously, but it is desirable to learn these from data as well. Just before the camera-ready version of this paper was due we developed a method for estimating the hyperparameters by putting a vague Gamma hyper-prior on each  $\alpha_A$  and sampled using Metropolis-Hastings with a sequence of increasingly narrow Gamma proposal distributions, producing results for each model that are as good or better than the best ones reported in Table 1.

The adaptor grammars presented here barely scratch the surface of the linguistically interesting models that can be expressed as Hierarchical Dirichlet Processes. The models of morphology presented here are particularly naive—they only capture regular concatenative morphology consisting of one paradigm class—which may partially explain why we obtained such poor results using morphology adaptor grammars. It’s straight forward to design an adaptor grammar that can capture a finite number of concatenative paradigm classes (Goldwater et al., 2006b; Johnson et al., 2007a). We’d like to learn the number of paradigm classes from the data, but doing this would probably require extending adaptor grammars to incorporate the kind of adaptive state-splitting found in the iHMM and iPCFG (Liang et al., 2007). There is no principled reason why this could not be done, i.e., why one could not design an HDP framework that simultaneously learns both the fragments (as in an adaptor grammar) and the states (as in an iHMM or iPCFG).

However, inference with these more complex models will probably itself become more complex. The MCMC sampler of Johnson et al. (2007a) used here is satisfactory for small and medium-sized problems, but it would be very useful to have more efficient inference procedures. It may be possible to adapt efficient split-merge samplers (Jain and Neal, 2007) and Variational Bayes methods (Teh et al., 2008) for DPs to adaptor grammars and other linguistic applications of HDPs.

## Acknowledgments

This research was funded by NSF awards 0544127 and 0631667.

## References

- N. Bernstein-Ratner. 1987. The phonology of parent-child speech. In K. Nelson and A. van Kleeck, editors, *Children's Language*, volume 6. Erlbaum, Hillsdale, NJ.
- Rens Bod. 1998. *Beyond grammar: an experience-based theory of language*. CSLI Publications, Stanford, California.
- Sharon Goldwater and Mark Johnson. 2005. Representational bias in unsupervised learning of syllable structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 112–119, Ann Arbor, Michigan, June. Association for Computational Linguistics.
- Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006a. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia, July. Association for Computational Linguistics.
- Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006b. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 459–466, Cambridge, MA. MIT Press.
- Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2007. Distributional cues to word boundaries: Context is important. In David Bamman, Tatiana Magnitskaia, and Colleen Zaller, editors, *Proceedings of the 31st Annual Boston University Conference on Language Development*, pages 239–250, Somerville, MA. Cascadilla Press.
- Sonia Jain and Radford M. Neal. 2007. Splitting and merging components of a nonconjugate dirichlet process mixture model. *Bayesian Analysis*, 2(3):445–472.
- Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007a. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April. Association for Computational Linguistics.
- Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. MIT Press, Cambridge, MA.
- Aravind Joshi. 2003. Tree adjoining grammars. In Ruslan Mikkov, editor, *The Oxford Handbook of Computational Linguistics*, pages 483–501. Oxford University Press, Oxford, England.
- Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697.
- Brian MacWhinney and Catherine Snow. 1985. The child language data exchange system. *Journal of Child Language*, 12:271–296.
- Karin Müller. 2001. Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.
- Karin Müller. 2002. Probabilistic context-free grammars for phonology. In *Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*, pages 70–80, Philadelphia.
- Andreas Stolcke. 1994. *Bayesian Learning of Probabilistic Language Models*. Ph.D. thesis, University of California, Berkeley.
- Y. W. Teh, M. Jordan, M. Beal, and D. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.
- Yee Whye Teh, Kenichi Kurihara, and Max Welling. 2008. Collapsed variational inference for hdp. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA.

# Inducing Gazetteers for Named Entity Recognition by Large-scale Clustering of Dependency Relations

**Jun'ichi Kazama**

Japan Advanced Institute of  
Science and Technology (JAIST),  
Asahidai 1-1, Nomi,  
Ishikawa, 923-1292 Japan  
kazama@jaist.ac.jp

**Kentaro Torisawa**

National Institute of Information and  
Communications Technology (NICT),  
3-5 Hikaridai, Seika-cho, Soraku-gun,  
Kyoto, 619-0289 Japan  
torisawa@nict.go.jp

## Abstract

We propose using large-scale clustering of dependency relations between verbs and multi-word nouns (MNs) to construct a gazetteer for named entity recognition (NER). Since dependency relations capture the semantics of MNs well, the MN clusters constructed by using dependency relations should serve as a good gazetteer. However, the high level of computational cost has prevented the use of clustering for constructing gazetteers. We parallelized a clustering algorithm based on expectation-maximization (EM) and thus enabled the construction of large-scale MN clusters. We demonstrated with the IREX dataset for the Japanese NER that using the constructed clusters as a gazetteer (*cluster gazetteer*) is an effective way of improving the accuracy of NER. Moreover, we demonstrate that the combination of the cluster gazetteer and a gazetteer extracted from Wikipedia, which is also useful for NER, can further improve the accuracy in several cases.

## 1 Introduction

*Gazetteers*, or entity dictionaries, are important for performing named entity recognition (NER) accurately. Since building and maintaining high-quality gazetteers by hand is very expensive, many methods have been proposed for automatic extraction of gazetteers from texts (Riloff and Jones, 1999; Thelen and Riloff, 2002; Etzioni et al., 2005; Shinzato et al., 2006; Talukdar et al., 2006; Nadeau et al., 2006).

Most studies using gazetteers for NER are based on the assumption that a gazetteer is a mapping

from a multi-word noun (MN)<sup>1</sup> to named entity categories such as “Tokyo Stock Exchange → {ORGANIZATION}”.<sup>2</sup> However, since the correspondence between the labels and the NE categories can be learned by tagging models, a gazetteer will be useful as long as it returns consistent labels even if those returned are not the NE categories. By changing the perspective in such a way, we can explore more broad classes of gazetteers. For example, we can use automatically extracted hyponymy relations (Hearst, 1992; Shinzato and Torisawa, 2004), or automatically induced MN clusters (Rooth et al., 1999; Torisawa, 2001).

For instance, Kazama and Torisawa (2007) used the hyponymy relations extracted from Wikipedia for the English NER, and reported improved accuracies with such a gazetteer.

We focused on the automatically induced clusters of multi-word nouns (MNs) as the source of gazetteers. We call the constructed gazetteers *cluster gazetteers*. In the context of tagging, there are several studies that utilized *word* clusters to prevent the data sparseness problem (Kazama et al., 2001; Miller et al., 2004). However, these methods cannot produce the MN clusters required for constructing gazetteers. In addition, the clustering methods used, such as HMMs and Brown’s algorithm (Brown et al., 1992), seem unable to adequately capture the semantics of MNs since they are based only on the information of adjacent words. We utilized richer

<sup>1</sup>We used the term, “multi-word”, to emphasize that a gazetteer includes not only one-word expressions but also multi-word expressions.

<sup>2</sup>Although several categories can be associated in general, we assume that only one category is associated.

syntactic/semantic structures, i.e., verb-MN dependencies to make clean MN clusters. Rooth et al. (1999) and Torisawa (2001) showed that the EM-based clustering using verb-MN dependencies can produce semantically clean MN clusters. However, the clustering algorithms, especially the EM-based algorithms, are computationally expensive. Therefore, performing the clustering with a vocabulary that is large enough to cover the many named entities required to improve the accuracy of NER is difficult. We enabled such large-scale clustering by parallelizing the clustering algorithm, and we demonstrate the usefulness of the gazetteer constructed.

We parallelized the algorithm of (Torisawa, 2001) using the Message Passing Interface (MPI), with the prime goal being to distribute parameters and thus enable clustering with a large vocabulary. Applying the parallelized clustering to a large set of dependencies collected from Web documents enabled us to construct gazetteers with up to 500,000 entries and 3,000 classes.

In our experiments, we used the IREX dataset (Sekine and Isahara, 2000) to demonstrate the usefulness of cluster gazetteers. We also compared the cluster gazetteers with the Wikipedia gazetteer constructed by following the method of (Kazama and Torisawa, 2007). The improvement was larger for the cluster gazetteer than for the Wikipedia gazetteer. We also investigated whether these gazetteers improve the accuracies further when they are used in combination. The experimental results indicated that the accuracy improved further in several cases and showed that these gazetteers complement each other.

The paper is organized as follows. In Section 2, we explain the construction of cluster gazetteers and its parallelization, along with a brief explanation of the construction of the Wikipedia gazetteer. In Section 3, we explain how to use these gazetteers as features in an NE tagger. Our experimental results are reported in Section 4.

## 2 Gazetteer Induction

### 2.1 Induction by MN Clustering

Assume we have a probabilistic model of a multi-word noun (MN) and its class:  $p(n, c) = p(n|c)p(c)$ , where  $n \in \mathcal{N}$  is an MN and  $c \in \mathcal{C}$  is a class. We can use this model to construct a gazetteer in several ways. The method we used in this study

constructs a gazetteer:  $n \rightarrow \operatorname{argmax}_c p(c|n)$ . This computation can be re-written by the Bayes rule as  $\operatorname{argmax}_c p(n|c)p(c)$  using  $p(n|c)$  and  $p(c)$ .

Note that we do not exclude non-NEs when we construct the gazetteer. We expect that tagging models (CRFs in our case) can learn an appropriate weight for each gazetteer match regardless of whether it is an NE or not.

### 2.2 EM-based Clustering using Dependency Relations

To learn  $p(n|c)$  and  $p(c)$  for Japanese, we use the EM-based clustering method presented by Torisawa (2001). This method assumes a probabilistic model of verb-MN dependencies with hidden semantic classes:<sup>3</sup>

$$p(v, r, n) = \sum_c p(\langle v, r \rangle | c) p(n|c) p(c), \quad (1)$$

where  $v \in \mathcal{V}$  is a verb and  $n \in \mathcal{N}$  is an MN that depends on verb  $v$  with relation  $r$ . A relation,  $r$ , is represented by Japanese postpositions attached to  $n$ . For example, from the following Japanese sentence, we extract the following dependency:  $v =$  飲む (drink),  $r =$  を ("wo" postposition),  $n =$  ビール (beer).

ビール (beer) を (wo) 飲む (drink) ( $\approx$  drink beer)

In the following, we let  $vt \equiv \langle v, r \rangle \in \mathcal{VT}$  for the simplicity of explanation.

To be precise, we attach various auxiliary verb suffixes, such as “れる (reru)”, which is for passivization, into  $v$ , since these greatly change the type of  $n$  in the dependent position. In addition, we also treated the MN-MN expressions, “ $MN_1$  の  $MN_2$ ” ( $\approx$  “ $MN_2$  of  $MN_1$ ”), as dependencies  $v = MN_2$ ,  $r =$  の,  $n = MN_1$ , since these expressions also characterize the dependent MNs well.

Given  $L$  training examples of verb-MN dependencies  $\{(vt_i, n_i, f_i)\}_{i=1}^L$ , where  $f_i$  is the number of dependency  $(vt_i, n_i)$  in a corpus, the EM-based clustering tries to find  $p(vt|c)$ ,  $p(n|c)$ , and  $p(c)$  that maximize the (log)-likelihood of the training examples:

$$LL(p) = \sum_i f_i \log \left( \sum_c p(vt_i|c) p(n_i|c) p(c) \right). \quad (2)$$

<sup>3</sup>This formulation is based on the formulation presented in Rooth et al. (1999) for English.

We iteratively update the probabilities using the EM algorithm. For the update procedures used, see Torisawa (2001).

The corpus we used for collecting dependencies was a large set (76 million) of Web documents, that were processed by a dependency parser, KNP (Kurohashi and Kawahara, 2005).<sup>4</sup> From this corpus, we extracted about 380 million dependencies of the form  $\{(vt_i, n_i, f_i)\}_i^L$ .

### 2.3 Parallelization for Large-scale Data

The disadvantage of the clustering algorithm described above is the computational costs. The space requirements are  $O(|\mathcal{VT}||\mathcal{C}| + |\mathcal{N}||\mathcal{C}| + |\mathcal{C}|)$  for storing the parameters,  $p(vt|c)$ ,  $p(n|c)$ , and  $p(c)$ <sup>5</sup>, plus  $O(L)$  for storing the training examples. The time complexity is mainly  $O(L \times |\mathcal{C}| \times I)$ , where  $I$  is the number of update iterations. The space requirements are the main limiting factor. Assume that a floating-point number consumes 8 bytes. With the setting,  $|\mathcal{N}| = 500,000$ ,  $|\mathcal{VT}| = 500,000$ , and  $|\mathcal{C}| = 3,000$ , the algorithm requires more than 44 GB for the parameters and 4 GB of memory for the training examples. A machine with more than 48 GB of memory is not widely available even today.

Therefore, we parallelized the clustering algorithm, to make it suitable for running on a cluster of PCs with a moderate amount of memory (e.g., 8 GB). First, we decided to store the training examples on a file since otherwise each node would need to store all the examples when we use the data splitting described below, and having every node consume 4 GB of memory is memory-consuming. Since the access to the training data is sequential, this does not slow down the execution when we use a buffering technique appropriately.<sup>6</sup>

We then split the matrix for the model parameters,  $p(n|c)$  and  $p(vt|c)$ , along with the class coordinate. That is, each cluster node is responsible for storing only a part of classes  $\mathcal{C}_l$ , i.e.,  $1/|P|$  of the parameter matrix, where  $P$  is the number of cluster nodes. This data splitting enables linear scalability of memory sizes. However, doing so complicates the update procedure and, in terms of execution speed, may

<sup>4</sup>**Acknowledgements:** This corpus was provided by Dr. Daisuke Kawahara of NICT.

<sup>5</sup>To be precise, we need two copies of these.

<sup>6</sup>Each node has a copy of the training data on a local disk.

**Algorithm 2.1:** Compute  $p(c_l|vt_i, n_i)$

```

localZ = 0, Z = 0
for  $c_l \in \mathcal{C}_l$  do
   $\begin{cases} d = p(vt_i|c)p(n_i|c)p(c) \\ p(c_l|vt_i, n_i) = d \\ localZ += d \end{cases}$ 
MPI_Allreduce(localZ, Z, 1, MPI.DOUBLE,
MPI.SUM, MPI.COMM_WORLD)
for  $c_l \in \mathcal{C}_l$  do  $p(c_l|vt_i, n_i) /= Z$ 

```

Figure 1: Parallelized inner-most routine of EM clustering algorithm. Each node executes this code in parallel.

offset the advantage of parallelization because each node needs to receive information about the classes that are not on the node in the inner-most routine of the update procedure.

The inner-most routine should compute:

$$p(c|vt_i, n_i) = p(vt_i|c)p(n_i|c)p(c)/Z, \quad (3)$$

for each class  $c$ , where  $Z = \sum_c p(vt_i|c)p(n_i|c)p(c)$  is a normalizing constant. However,  $Z$  cannot be calculated without knowing the results of other cluster nodes. Thus, if we use MPI for parallelization, the parallelized version of this routine should resemble the algorithm shown in Figure 1. This routine first computes  $p(vt_i|c_l)p(n_i|c_l)p(c_l)$  for each  $c_l \in \mathcal{C}_l$ , and stores the sum of these values as  $localZ$ . The routine uses an MPI function, MPI\_Allreduce, to sum up  $localZ$  of the all cluster nodes and to set  $Z$  with the resulting sum. We can compute  $p(c_l|vt_i, n_i)$  by using this  $Z$  to normalize the value. Although the above is the essence of our parallelization, invoking MPI\_Allreduce in the inner-most loop is very expensive because the communication setup is not so cheap. Therefore, our implementation calculates  $p(c_l|vt_i, n_i)$  in batches of  $B$  examples and calls MPI\_Allreduce at every  $B$  examples.<sup>7</sup> We used a value of  $B = 4,096$  in this study.

By using this parallelization, we successfully performed the clustering with  $|\mathcal{N}| = 500,000$ ,  $|\mathcal{VT}| = 500,000$ ,  $|\mathcal{C}| = 3,000$ , and  $I = 150$ , on 8 cluster nodes with a 2.6 GHz Opteron processor and 8 GB of memory. This clustering took about a week. To our knowledge, no one else has performed EM-based clustering of this type on this scale. The resulting MN clusters are shown in Figure 2. In terms of speed, our experiments are still at a preliminary

<sup>7</sup>MPI\_Allreduce can also take array arguments and apply the operation to each element of the array in one call.

Class 791	Class 2760
ウインダム (WINDOM)	マリン/スタジアム (Chiba Marine Stadium [abb.])
カムリ (CAMRY)	大阪/ドーム (Osaka Dome)
ディアマンテ (DIAMANTE)	ナゴド (Nagoya Dome [abb.])
オデッセイ (ODYSSEY)	福岡/ドーム (Fukuoka Dome)
インスパイア (INSPIRE)	大阪/球場 (Osaka Stadium)
スイフト (SWIFT)	ハマ/スタ (Yokohama Stadium [abb.])

Figure 2: Clean MN clusters with named entity entries (Left: car brand names. Right: stadium names). Names are sorted on the basis of  $p(c|n)$ . Stadium names are examples of multi-word nouns (word boundaries are indicated by “/”) and also include abbreviated expressions (marked by [abb.] ).

stage. We have observed 5 times faster execution, when using 8 cluster nodes with a relatively small setting,  $|\mathcal{N}| = |\mathcal{VT}| = 50,000$ ,  $|\mathcal{C}| = 2,000$ .

## 2.4 Induction from Wikipedia

Defining sentences in a dictionary or an encyclopedia have long been used as a source of hyponymy relations (Tsurumaru et al., 1991; Herbelot and Copestake, 2006).

Kazama and Torisawa (2007) extracted hyponymy relations from the first sentences (i.e., defining sentences) of Wikipedia articles and then used them as a gazetteer for NER. We used this method to construct the Wikipedia gazetteer.

The method described by Kazama and Torisawa (2007) is to first extract the first (base) noun phrase after the first “is”, “was”, “are”, or “were” in the first sentence of a Wikipedia article. The last word in the noun phrase is then extracted and becomes the hypernym of the entity described by the article. For example, from the following defining sentence, it extracts “guitarist” as the hypernym for “Jimi Hendrix”.

Jimi Hendrix (November 27, 1942) was an American guitarist, singer and songwriter.

The second noun phrase is used when the first noun phrase ends with “one”, “kind”, “sort”, or “type”, or it ended with “name” followed by “of”. This rule is for treating expressions like “... is one of the landlocked countries.” By applying this method of extraction to all the articles in Wikipedia, we

	# instances
page titles processed	550,832
articles found (found by redirection)	547,779 (189,222)
first sentences found	545,577
hypernyms extracted	482,599

Table 1: Wikipedia gazetteer extraction

construct a gazetteer that maps an MN (a title of a Wikipedia article) to its hypernym.<sup>8</sup> When the hypernym extraction failed, a special hypernym symbol, e.g., “UNK”, was used.

We modified this method for Japanese. After pre-processing the first sentence of an article using a morphological analyzer, MeCab<sup>9</sup>, we extracted the *last* noun after the appearance of Japanese postposition “は (wa)” ( $\approx$  “is”). As in the English case, we also refrained from extracting expressions corresponding to “one of” and so on.

From the Japanese Wikipedia entries of April 10, 2007, we extracted 550,832 gazetteer entries (482,599 entries have hypernyms other than UNK). Various statistics for this extraction are shown in Table 1. The number of distinct hypernyms in the gazetteer was 12,786. Although this Wikipedia gazetteer is much smaller than the English version used by Kazama and Torisawa (2007) that has over 2,000,000 entries, it is the largest gazetteer that can be freely used for Japanese NER. Our experimental results show that this Wikipedia gazetteer can be used to improve the accuracy of Japanese NER.

## 3 Using Gazetteers as Features of NER

Since Japanese has no spaces between words, there are several choices for the token unit used in NER. Asahara and Motsumoto (2003) proposed using characters instead of morphemes as the unit to alleviate the effect of segmentation errors in morphological analysis and we also used their character-based method. The NER task is then treated as a tagging task, which assigns IOB tags to each character in a sentence.<sup>10</sup> We use Conditional Random Fields (CRFs) (Lafferty et al., 2001) to perform this tagging.

The information of a gazetteer is incorporated

<sup>8</sup>They handled “redirections” as well by following redirection links and extracting a hypernym from the article reached.

<sup>9</sup><http://mecab.sourceforge.net>

<sup>10</sup>Precisely, we use IOB2 tags.

<i>ch</i>	に	ソ	ニ	一	が	開	発	...
match	O	B	I	I	O	O	O	...
(w/ class)	O	B-会社	I-会社	I-会社	O	O	O	...

Figure 3: Gazetteer features for Japanese NER. Here, ‘ソニ一’ means ‘‘SONY’’, ‘‘会社’’ means ‘‘company’’, and ‘‘開発’’ means ‘‘to develop’’.

as features in a CRF-based NE tagger. We follow the method used by Kazama and Torisawa (2007), which encodes the matching with a gazetteer entity using IOB tags, with the modification for Japanese. They describe using two types of gazetteer features. The first is a matching-only feature, which uses bare IOB tags to encode only matching information. The second uses IOB tags that are augmented with classes (e.g., B-country and I-country).<sup>11</sup> When there are several possibilities for making a match, the left-most longest match is selected. The small differences from their work are: (1) We used characters as the unit as we described above, (2) While Kazama and Torisawa (2007) checked only the word sequences that start with a capitalized word and thus exploited the characteristics of English language, we checked the matching at every character, (3) We used a TRIE to make the look-up efficient.

The output of gazetteer features for Japanese NER are thus as those shown in Figure 3. These annotated IOB tags can be used in the same way as other features in a CRF tagger.

## 4 Experiments

### 4.1 Data

We used the CRL NE dataset provided in the IREX competition (Sekine and Isahara, 2000). In the dataset, 1,174 newspaper articles are annotated with 8 NE categories: ARTIFACT, DATE, LOCATION, MONEY, ORGANIZATION, PERCENT, PERSON, and TIME.<sup>12</sup> We converted the data into the CoNLL 2003 format, i.e., each row corresponds to a character in this case. We obtained 11,892 sentences<sup>13</sup> with 18,677 named entities. We split this data into the training set (9,000 sentences), the de-

<sup>11</sup>Here, we call the value returned by a gazetteer a ‘‘class’’. Features are not output when the returned class is UNK in the case of the Wikipedia gazetteer. We did not observe any significant change if we also used UNK.

<sup>12</sup>We ignored OPTIONAL category.

<sup>13</sup>This number includes the number of -DOCSTART- tokens in CoNLL 2003 format.

Name	Description
ch	character itself
ct	character type: uppercase alphabet, lowercase alphabet, katakana, hiragana, Chinese characters, numbers, numbers in Chinese characters, and spaces
m_mo	bare IOB tag indicating boundaries of morphemes
m_mm	IOB tag augmented by morpheme string, indicating boundaries and morphemes
m_mp	IOB tag augmented by morpheme type, indicating boundaries and morpheme types (POSs)
bm	bare IOB tag indicating ‘‘bunsetsu’’ boundaries (Bunsetsu is a basic unit in Japanese and usually contains content words followed by function words such as postpositions)
bi	bunsetsu-inner feature. See (Nakano and Hirai, 2004).
bp	adjacent-bunsetsu feature. See (Nakano and Hirai, 2004).
bh	head-of-bunsetsu features. See (Nakano and Hirai, 2004).

Table 2: Atomic features used in baseline model.

velopment set (1,446 sentences), and the testing set (1,446 sentences).

### 4.2 Baseline Model

We extracted the atomic features listed in Table 2 at each character for our baseline model. Though there may be slight differences, these features are based on the standard ones proposed and used in previous studies on Japanese NER such as those by Asahara and Motsumoto (2003), Nakano and Hirai (2004), and Yamada (2007). We used MeCab as a morphological analyzer and CaboCha<sup>14</sup> (Kudo and Matsumoto, 2002) as the dependency parser to find the boundaries of the bunsetsu. We generated the node and the edge features of a CRF model as described in Table 3 using these atomic features.

### 4.3 Training

To train CRF models, we used Taku Kudo’s CRF++ (ver. 0.44)<sup>15</sup> with some modifications.<sup>16</sup> We

<sup>14</sup><http://chasen.org/~taku/software/CaboCha>

<sup>15</sup><http://chasen.org/~taku/software/CRF++>

<sup>16</sup>We implemented scaling, which is similar to that for HMMs (Rabiner, 1989), in the forward-backward phase and replaced the optimization module in the original package with the

<b>Node features:</b>
$\{"" , x_{-2}, x_{-1}, x_0, x_{+1}, x_{+2}\} \times y_0$ where $x = \text{ch, ct, m\_mm, m\_mo, m\_mp, bi, bp, and bh}$
<b>Edge features:</b>
$\{"" , x_{-1}, x_0, x_{+1}\} \times y_{-1} \times y_0$ where $x = \text{ch, ct, and m\_mp}$
<b>Bigram node features:</b>
$\{x_{-2}x_{-1}, x_{-1}x_0, x_0x_{+1}\} \times y_0$ $x = \text{ch, ct, m\_mo, m\_mp, bm, bi, bp, and bh}$

Table 3: Baseline features. Value of node feature is determined from current tag,  $y_0$ , and surface feature (combination of atomic features in Table 2). Value of edge feature is determined by previous tag,  $y_{-1}$ , current tag,  $y_0$ , and surface feature. Subscripts indicate relative position from current character.

used Gaussian regularization to prevent overfitting. The parameter of the Gaussian,  $\sigma^2$ , was tuned using the development set. We tested 10 points:  $\{0.64, 1.28, 2.56, 5.12, \dots, 163.84, 327.68\}$ . We stopped training when the relative change in the log-likelihood became less than a pre-defined threshold, 0.0001. Throughout the experiments, we omitted the features whose surface part described in Table 3 occurred less than twice in the training corpus.

#### 4.4 Effect of Gazetteer Features

We investigated the effect of the cluster gazetteer described in Section 2.1 and the Wikipedia gazetteer described in Section 2.4, by adding each gazetteer to the baseline model. We added the matching-only and the class-augmented features, and we generated the node and the edge features in Table 3.<sup>17</sup> For the cluster gazetteer, we made several gazetteers that had different vocabulary sizes and numbers of classes. The number of clustering iterations was 150 and the initial parameters were set randomly with a Dirichlet distribution ( $\alpha_i = 1.0$ ).

The statistics of each gazetteer are summarized in Table 4. The number of entries in a gazetteer is given by “# entries”, and “# matches” is the number of matches that were output for the training set. We define “# e-matches” as the number of matches that also match a boundary of a named entity in the training set, and “# optimal” as the optimal number of “# e-matches” that can be achieved when we know the

oracle of entity boundaries. Note that this cannot be realized because our matching uses the left-most longest heuristics. We define “pre.” as the precision of the output matches (i.e., # e-matches/# matches), and “rec.” as the recall (i.e., # e-matches/# NEs). Here, # NEs = 14,056. Finally, “opt.” is the optimal recall (i.e., # optimal/# NEs). “# classes” is the number of distinct classes in a gazetteer, and “# used” is the number of classes that were output for the training set. Gazetteers are as follows: “wikip(m)” is the Wikipedia gazetteer (matching only), and “wikip(c)” is the Wikipedia gazetteer (with class-augmentation). A cluster gazetteer, which is constructed by the clustering with  $|\mathcal{N}| = |\mathcal{VT}| = X \times 1,000$  and  $|\mathcal{C}| = Y \times 1,000$ , is indicated by “cXk-Yk”. Note that “# entries” is slightly smaller than the vocabulary size since we removed some duplications during the conversion to a TRIE.

These gazetteers cover 40 - 50% of the named entities, and the cluster gazetteers have relatively wider coverage than the Wikipedia gazetteer has. The precisions are very low because there are many erroneous matches, e.g., with a entries for a hiragana character.<sup>18</sup> Although this seems to be a serious problem, removing such one-character entries does not affect the accuracy, and in fact, makes it worsen slightly. We think this shows one of the strengths of machine learning methods such as CRFs. We can also see that our current matching method is not an optimal one. For example, 16% of the matches were lost as a result of using our left-most longest heuristics for the case of the c500k-2k gazetteer.

A comparison of the effect of these gazetteers is shown in Table 5. The performance is measured by the F-measure. First, the Wikipedia gazetteer improved the accuracy as expected, i.e., it reproduced the result of Kazama and Torisawa (2007) for Japanese NER. The improvement for the testing set was 1.08 points. Second, all the tested cluster gazetteers improved the accuracy. The largest improvement was 1.55 points with the c300k-3k gazetteer. This was larger than that of the Wikipedia gazetteer. The results for c300k-Yk gazetteers show a peak of the improvement at some number of clusters. In this case,  $|\mathcal{C}| = 3,000$  achieved the best improvement. The results of cXk-2k gazetteers in-

LMVM optimizer of TAO (version 1.9) (Benson et al., 2007)

<sup>17</sup>Bigram node features were not used for gazetteer features.

<sup>18</sup>Wikipedia contains articles explaining each hiragana character, e.g., “あ is a hiragana character”.



Name	# entries	# matches	# e-matches	# optimal	pre. (%)	rec. (%)	opt. rec. (%)	# classes	# used
wikip(m)	550,054	225,607	6,804	7,602	3.02	48.4	54.1	N/A	N/A
wikip(c)	550,054	189,029	5,441	6,064	2.88	38.7	43.1	12,786	1,708
c100k-2k	99,671	193,897	6,822	8,233	3.52	48.5	58.6	2,000	1,910
c300k-2k	295,695	178,220	7,377	9,436	4.14	52.5	67.1	2,000	1,973
c300k-1k	↑	↑	↑	↑	↑	↑	↑	1,000	982
c300k-3k	↑	↑	↑	↑	↑	↑	↑	3,000	2,848
c300k-4k	↑	↑	↑	↑	↑	↑	↑	4,000	3,681
c500k-2k	497,101	174,482	7,470	9,798	4.28	53.1	69.7	2,000	1,951
c500k-3k	↑	↑	↑	↑	↑	↑	↑	3,000	2,854

Table 4: Statistics of various gazetteers.

Model	F (dev.)	F (test.)	best $\sigma^2$
baseline	87.23	87.42	20.48
+wikip	<b>87.60</b>	<b>88.50</b>	2.56
+c300k-1k	88.74	87.98	40.96
+c300k-2k	88.75	88.01	163.84
+c300k-3k	<b>89.12</b>	<b>88.97</b>	20.48
+c300k-4k	88.99	88.40	327.68
+c100k-2k	88.15	88.06	20.48
+c500k-2k	88.80	88.12	40.96
+c500k-3k	88.75	88.03	20.48

Table 5: Comparison of gazetteer features.

Model	F
(Asahara and Motsumoto, 2003)	87.21
(Nakano and Hirai, 2004)	89.03
(Yamada, 2007)	88.33
(Sasano and Kurohashi, 2008)	89.40
proposed (baseline)	87.62
proposed (+wikip)	88.14
proposed (+c300k-3k)	88.45
proposed (+c500k-2k)	88.41
proposed (+wikip+c300k-3k)	<b>88.93</b>
proposed (+wikip+c500k-2k)	88.71

Table 7: Comparison with previous studies

Model	F (dev.)	F (test.)	best $\sigma^2$
+wikip+c300k-1k	88.65	*89.32	0.64
+wikip+c300k-2k	*89.22	*89.13	10.24
+wikip+c300k-3k	88.69	*89.62	40.96
+wikip+c300k-4k	88.67	*89.19	40.96
+wikip+c500k-2k	* <b>89.26</b>	* <b>89.19</b>	2.56
+wikip+c500k-3k	*88.80	*88.60	10.24

Table 6: Effect of combination. Figures with \* mean that accuracy was improved by combining gazetteers.

dicating that the larger a gazetteer is, the larger the improvement. However, the accuracies of the c300k-3k and c500k-3k gazetteers seem to contradict this tendency. It might be caused by the accidental low quality of the clustering that results from random initialization. We need to investigate this further.

#### 4.5 Effect of Combining the Cluster and the Wikipedia Gazetteers

We have observed that using the cluster gazetteer and the Wikipedia one improves the accuracy of Japanese NER. The next question is whether these gazetteers improve the accuracy further when they are used together. The accuracies of models that use the Wikipedia gazetteer and one of the cluster gazetteers at the same time are shown in Table 6. The accuracy was improved in most cases. How-

ever, there were some cases where the accuracy for the development set was degraded. Therefore, we should state at this point that while the benefit of combining these gazetteers is not consistent in a strict sense, it seems to exist. The best performance,  $F = 89.26$  (dev.) /  $89.19$  (test.), was achieved when we combined the Wikipedia gazetteer and the cluster gazetteer, c500k-2k. This means that there was a 1.77-point improvement from the baseline for the testing set.

## 5 Comparison with Previous Studies

Since many previous studies on Japanese NER used 5-fold cross validation for the IREX dataset, we also performed it for some of our models that had the best  $\sigma^2$  found in the previous experiments. The results are listed in Table 7 with references to the results of recent studies. These results not only reconfirmed the effects of the gazetteer features shown in the previous experiments, but they also showed that our best model is comparable to the state-of-the-art models. The system recently proposed by Sasano and Kurohashi (2008) is currently the best system for the IREX dataset. It uses many structural features that are not used in our model. Incorporating

such features might improve our model further.

## 6 Related Work and Discussion

There are several studies that used automatically extracted gazetteers for NER (Shinzato et al., 2006; Talukdar et al., 2006; Nadeau et al., 2006; Kazama and Torisawa, 2007). Most of the methods (Shinzato et al., 2006; Talukdar et al., 2006; Nadeau et al., 2006) are oriented at the NE category. They extracted a gazetteer for each NE category and utilized it in a NE tagger. On the other hand, Kazama and Torisawa (2007) extracted hyponymy relations, which are independent of the NE categories, from Wikipedia and utilized it as a gazetteer. The effectiveness of this method was demonstrated for Japanese NER as well by this study.

Inducing features for taggers by clustering has been tried by several researchers (Kazama et al., 2001; Miller et al., 2004). They constructed word clusters by using HMMs or Brown’s clustering algorithm (Brown et al., 1992), which utilize only information from neighboring words. This study, on the other hand, utilized MN clustering based on verb-MN dependencies (Rooth et al., 1999; Torisawa, 2001). We showed that gazetteers created by using such richer semantic/syntactic structures improves the accuracy for NER.

The size of the gazetteers is also a novel point of this study. The previous studies, with the exception of Kazama and Torisawa (2007), used smaller gazetteers than ours. Shinzato et al. (2006) constructed gazetteers with about 100,000 entries in total for the “restaurant” domain; Talukdar et al. (2006) used gazetteers with about 120,000 entries in total, and Nadeau et al. (2006) used gazetteers with about 85,000 entries in total. By parallelizing the clustering algorithm, we successfully constructed a cluster gazetteer with up to 500,000 entries from a large amount of dependency relations in Web documents. To our knowledge, no one else has performed this type of clustering on such a large scale. Wikipedia also produced a large gazetteer of more than 550,000 entries. However, comparing these gazetteers and ours precisely is difficult at this point because the detailed information such as the precision and the recall of these gazetteers were not reported.<sup>19</sup> Recently, Inui et al. (2007) investi-

<sup>19</sup>Shinzato et al. (2006) reported some useful statistics about

gated the relation between the size and the quality of a gazetteer and its effect. We think this is one of the important directions of future research.

Parallelization has recently regained attention in the machine learning community because of the need for learning from very large sets of data. Chu et al. (2006) presented the MapReduce framework for a wide range of machine learning algorithms, including the EM algorithm. Newman et al. (2007) presented parallelized Latent Dirichlet Allocation (LDA). However, these studies focus on the distribution of the training examples and relevant computation, and ignore the need that we found for the distribution of model parameters. The exception, which we noticed recently, is a study by Wolfe et al. (2007), which describes how each node stores only those parameters relevant to the training data on each node. However, some parameters need to be duplicated and thus their method is less efficient than ours in terms of memory usage.

We used the left-most longest heuristics to find the matching gazetteer entries. However, as shown in Table 4 this is not an optimal method. We need more sophisticated matching methods that can handle multiple matching possibilities. Using models such as Semi-Markov CRFs (Sarawagi and Cohen, 2004), which handle the features on overlapping regions, is one possible direction. However, even if we utilize the current gazetteers optimally, the coverage is upper bounded at 70%. To cover most of the named entities in the data, we need much larger gazetteers. A straightforward approach is to increase the number of Web documents used for the MN clustering and to use larger vocabularies.

## 7 Conclusion

We demonstrated that a gazetteer obtained by clustering verb-MN dependencies is a useful feature for a Japanese NER. In addition, we demonstrated that using the cluster gazetteer and the gazetteer extracted from Wikipedia (also shown to be useful) can together further improves the accuracy in several cases. Future work will be to refine the matching method and to construct even larger gazetteers.

their gazetteers.

## References

- M. Asahara and Y. Motsumoto. 2003. Japanese named entity extraction with redundant morphological analysis.
- S. Benson, L. C. McInnes, J. Moré, T. Munson, and J. Sarich. 2007. TAO user manual (revision 1.9). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory. <http://www.mcs.anl.gov/tao>.
- P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.
- C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. 2006. Map-reduce for machine learning on multicore. In *NIPS 2006*.
- O. Etzioni, M. Cafarella, D. Downey, A. M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the Web – an experimental study. *Artificial Intelligence Journal*.
- M. A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th International Conference on Computational Linguistics*, pages 539–545.
- A. Herbelot and A. Copestake. 2006. Acquiring ontological relationships from Wikipedia using RMRS. In *Workshop on Web Content Mining with Human Language Technologies ISWC06*.
- T. Inui, K. Murakami, T. Hashimoto, K. Utsumi, and M. Ishikawa. 2007. A study on using gazetteers for organization name recognition. In *IPSJ SIG Technical Report 2007-NL-182 (in Japanese)*.
- J. Kazama and K. Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *EMNLP-CoNLL 2007*.
- J. Kazama, Y. Miyao, and J. Tsujii. 2001. A maximum entropy tagger with unsupervised hidden Markov models. In *NLPRS 2001*.
- T. Kudo and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *CoNLL 2002*.
- S. Kurohashi and D. Kawahara. 2005. KNP (Kurohashi-Nagao parser) 2.0 users manual.
- J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*.
- S. Miller, J. Guinness, and A. Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL04*.
- D. Nadeau, Peter D. Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *19th Canadian Conference on Artificial Intelligence*.
- K. Nakano and Y. Hirai. 2004. Japanese named entity extraction with bunsetsu features. *IPSJ Journal (in Japanese)*.
- D. Newman, A. Asuncion, P. Smyth, and M. Welling. 2007. Distributed inference for latent dirichlet allocation. In *NIPS 2007*.
- L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.
- E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *16th National Conference on Artificial Intelligence (AAAI-99)*.
- M. Rooth, S. Riezler, D. Presher, G. Carroll, and F. Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering.
- S. Sarawagi and W. W. Cohen. 2004. Semi-Markov random fields for information extraction. In *NIPS 2004*.
- R. Sasano and S. Kurohashi. 2008. Japanese named entity recognition using structural natural language processing. In *IJCNLP 2008*.
- S. Sekine and H. Isahara. 2000. IREX: IR and IE evaluation project in Japanese. In *IREX 2000*.
- K. Shinzato and K. Torisawa. 2004. Acquiring hyponymy relations from Web documents. In *HLT-NAACL 2004*.
- K. Shinzato, S. Sekine, N. Yoshinaga, and K. Torisawa. 2006. Constructing dictionaries for named entity recognition on specific domains from the Web. In *Web Content Mining with Human Language Technologies Workshop on the 5th International Semantic Web*.
- P. P. Talukdar, T. Brants, M. Liberman, and F. Pereira. 2006. A context pattern induction method for named entity extraction. In *CoNLL 2006*.
- M. Thelen and E. Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern context. In *EMNLP 2002*.
- K. Torisawa. 2001. An unsupervised method for canonicalization of Japanese postpositions. In *NLPRS 2001*.
- H. Tsurumaru, K. Takeshita, K. Iami, T. Yanagawa, and S. Yoshida. 1991. An approach to thesaurus construction from Japanese language dictionary. In *IPSJ SIG Notes Natural Language vol.83-16, (in Japanese)*.
- J. Wolfe, A. Haghighi, and D. Klein. 2007. Fully distributed EM for very large datasets. In *NIPS Workshop on Efficient Machine Learning*.
- H. Yamada. 2007. Shift-reduce chunking for Japanese named entity extraction. In *IPSJ SIG Technical Report 2007-NL-179*.

# Evaluating *Roget's* Thesauri

**Alistair Kennedy**

School of Information Technology  
and Engineering  
University of Ottawa  
Ottawa, Ontario, Canada  
akennedy@site.uottawa.ca

**Stan Szpakowicz**

School of Information Technology  
and Engineering  
University of Ottawa  
Ottawa, Ontario, Canada  
and  
Institute of Computer Science  
Polish Academy of Sciences  
Warsaw, Poland  
szpak@site.uottawa.ca

## Abstract

*Roget's* Thesaurus has gone through many revisions since it was first published 150 years ago. But how do these revisions affect *Roget's* usefulness for NLP? We examine the differences in content between the 1911 and 1987 versions of *Roget's*, and we test both versions with each other and *WordNet* on problems such as synonym identification and word relatedness. We also present a novel method for measuring sentence relatedness that can be implemented in either version of *Roget's* or in *WordNet*. Although the 1987 version of the Thesaurus is better, we show that the 1911 version performs surprisingly well and that often the differences between the versions of *Roget's* and *WordNet* are not statistically significant. We hope that this work will encourage others to use the 1911 *Roget's* Thesaurus in NLP tasks.

## 1 Introduction

*Roget's* Thesaurus, first introduced over 150 years ago, has gone through many revisions to reach its current state. We compare two versions, the 1987 and 1911 editions of the Thesaurus with each other and with *WordNet* 3.0. *Roget's* Thesaurus has a unique structure, quite different from *WordNet*, of which the NLP community has yet to take full advantage. In this paper we demonstrate that although the 1911 version of the Thesaurus is very old, it can give results comparable to systems that use *WordNet* or newer versions of *Roget's* Thesaurus.

The main motivation for working with the 1911 Thesaurus instead of newer versions is that it is in

the public domain, along with related NLP-oriented software packages. For applications that call for an NLP-friendly thesaurus, *WordNet* has become the de-facto standard. Although *WordNet* is a fine resource, we believe that ignoring other thesauri is a serious oversight. We show on three applications how useful the 1911 Thesaurus is. We ran the well-established tasks of determining semantic relatedness of pairs of terms and identifying synonyms (Jarmasz and Szpakowicz, 2004). We also proposed a new method of representing the meaning of sentences or other short texts using either *WordNet* or *Roget's* Thesaurus, and tested it on the data set provided by Li et al. (2006). We hope that this work will encourage others to use *Roget's* Thesaurus in their own NLP tasks.

Previous research on the 1987 version of *Roget's* Thesaurus includes work of Jarmasz and Szpakowicz (2004). They propose a method of determining semantic relatedness between pairs of terms. Terms that appear closer together in the Thesaurus get higher weights than those farther apart. The experiments aimed at identifying synonyms using a modified version of the proposed semantic similarity function. Similar experiments were carried out using *WordNet* in combination with a variety of semantic relatedness functions. *Roget's* Thesaurus was found generally to outperform *WordNet* on these problems. We have run similar experiments using the 1911Thesaurus.

Lexical chains have also been developed using the 1987 *Roget's* Thesaurus (Jarmasz and Szpakowicz, 2003). The procedure maps words in a text to the Head (a *Roget's* concept) from which they are most likely to come. Although we did not experiment

with lexical chains here, they were an inspiration for our sentence relatedness function.

*Roget's* Thesaurus does not explicitly label the relations between its terms, as *WordNet* does. Instead, it groups terms together with implied relations. Kennedy and Szpakowicz (2007) show how disambiguating one of these relations, hypernymy, can help improve the semantic similarity functions in (Jarmasz and Szpakowicz, 2004). These hypernym relations were also put towards solving analogy questions.

This is not the first time the 1911 version of *Roget's* Thesaurus has been used in NLP research. Cassidy (2000) used it to build the semantic network FACTOTUM. This required significant (manual) restructuring, so FACTOTUM cannot really be considered a true version of *Roget's* Thesaurus.

The 1987 data come from *Penguin's Roget's Thesaurus* (Kirkpatrick, 1987). The 1911 version is available from Project Gutenberg<sup>1</sup>. We use *WordNet* 3.0, the latest version (Fellbaum, 1998). In the experiments we present here, we worked with an interface to *Roget's* Thesaurus implemented in Java 5.0<sup>2</sup>. It is built around a large index which stores the location in the thesaurus of each word or phrase; the system individually indexes all words within each phrase, as well as the phrase itself. This was shown to improve results in a few applications, which we will discuss later in the paper.

## 2 Content comparison of the 1911 and 1987 Thesauri

Although the 1987 and 1911 Thesauri are very similar in structure, there are a few differences, among them, the number of levels and the number of parts-of-speech represented. For example, the 1911 version contains some pronouns as well as more sections dedicated to phrases.

There are nine levels in *Roget's* Thesaurus hierarchy, from Class down to Word. We show them in Table 1 along with the counts of instances of each level. An example of a Class in the 1911 Thesaurus is “Words Expressing Abstract Relations”, a Section in that Class is “Quantity” with a Subsection “Comparative Quantity”. Heads can be thought of as the heart of the Thesaurus because it is at this level that

<sup>1</sup><http://www.gutenberg.org/ebooks/22>

<sup>2</sup><http://rogets.site.uottawa.ca/>

Hierarchy	1911	1987
Class	8	8
Section	39	39
Subsection	97	95
Head Group	625	596
Head	1044	990
Part-of-speech	3934	3220
Paragraph	10244	6443
Semicolon Group	43196	59915
Total Words	98924	225124
Unique Words	59768	100470

Table 1: Frequencies of each level of the hierarchy in the 1911 and 1987 Thesauri.

the lexical material, organized into approximately a thousand concepts, resides. Head Groups often pair up opposites, for example Head #1 “Existence” and Head #2 “Nonexistence” are found in the same Head Group in both versions of the Thesaurus. Terms in the Thesaurus may be labelled with cross-references to other words in different Heads. We did not use these references in our experiments.

The part-of-speech level is a little confusing, since clearly no such grouping contains an exhaustive list of all nouns, all verbs etc. We will write “POS” to indicate a structure in *Roget's* and “part-of-speech” to indicate the word category in general. The four main parts-of-speech represented in a POS are nouns, verbs, adjectives and adverbs. Interjections are also included in both the 1911 and 1987 thesauri; they are usually phrases followed by an exclamation mark, such as “for God’s sake!” and “pshaw!”. The Paragraph and Semicolon Group are not given names, but can often be represented by the first word.

The 1911 version also contains phrases (mostly quotations), prefixes and pronouns. There are only three prefixes – “tri-”, “tris-”, “laevo-” – and six pronouns – “he”, “him”, “his”, “she”, “her”, “hers”.

Table 2 shows the frequency of paragraphs, semicolon groups and both total and unique words in a given type of POS. Many terms occur both in the 1911 and 1987 Thesauri, but many more are unique to either. Surprisingly, quite a few 1911 terms do not appear in the 1987 data, as shown in Table 3; many of them may have been considered obsolete and thus dropped from the 1987 version. For example “ingrafted” appears in the same semicolon group as

POS	Paragraph		Semicolon Grp	
	1911	1987	1911	1987
Noun	4495	2884	19215	31174
Verb	2402	1499	10838	13958
Adjective	2080	1501	9097	12893
Adverb	594	499	2028	1825
Interjection	108	60	149	65
Phrase	561	0	1865	0
	Total Word		Unique Words	
	1911	1987	1911	1987
Noun	46308	114473	29793	56187
Verb	25295	55724	15150	24616
Adjective	20447	48802	12739	21614
Adverb	4039	5720	3016	4144
Interjection	598	405	484	383
Phrase	2228	0	2038	0

Table 2: Frequencies of paragraphs, semicolon groups, total words and unique words by their part of speech; we omitted prefixes and pronouns.

POS	Both	Only 1911	Only 1987
All	35343	24425	65127
N.	18685	11108	37502
Vb.	8618	6532	15998
Adj.	8584	4155	13030
Adv.	1684	1332	2460
Int.	68	416	315
Phr.	0	2038	0

Table 3: Frequencies of terms in either the 1911 or 1987 Thesaurus, and in both; we omitted prefixes and pronouns.

“implanted” in the older but not the newer version. Some mismatches may be due to small changes in spelling, for example, “Nirvana” is capitalized in the 1911 version, but not in the 1987 version.

The lexical data in Project Gutenberg’s 1911 *Roget’s* appear to have been somewhat added to. For example, the citation “Go ahead, make my day!” from the 1971 movie *Dirty Harry* appears twice (in Heads #715-Defiance and #761-Prohibition) within the *Phrase* POS. It is not clear to what extent new terms have been added to the original 1911 *Roget’s* Thesaurus, or what the criteria for adding such new elements could have been.

In the end, there are many differences between the 1987 and 1911 *Roget’s* Thesauri, primarily in con-

tent rather than in structure. The 1987 Thesaurus is largely an expansion of the 1911 version, with three POSs (phrases, pronouns and prefixes) removed.

### 3 Comparison on applications

In this section we consider how the two versions of *Roget’s* Thesaurus and *WordNet* perform in three applications – measuring word relatedness, synonym identification, and sentence relatedness.

#### 3.1 Word relatedness

Relatedness can be measured by the closeness of the words or phrases – henceforth referred to as *terms* – in the structure of the thesaurus. Two terms in the same semicolon group score 16, in the same paragraph – 14, and so on (Jarmasz and Szpakowicz, 2004). The score is 0 if the terms appear in different classes, or if either is missing. Pairs of terms get higher scores for being closer together. When there are multiple senses of two terms  $A$  and  $B$ , we want to select senses  $a \in A$  and  $b \in B$  that maximize the relatedness score. We define a distance function:

$$semDist(A, B) = \max_{a \in A, b \in B} 2 * (depth(lca(a, b)))$$

*lca* is the *lowest common ancestor* and *depth* is the depth in the *Roget’s* hierarchy; a Class has depth 0, Section 1, ..., Semicolon Group 8. If we think of the function as counting edges between concepts in the *Roget’s* hierarchy, then it could also be written as:

$$semDist(A, B) = \max_{a \in A, b \in B} 16 - edgesBetween(a, b)$$

We do not count links between words in the same semicolon group, so in effect these methods find distances between semicolon groups, that is to say, these two functions will give the same results.

The 1911 and 1987 Thesauri were compared with *WordNet* 3.0 on the three data sets containing pairs of words with manually assigned similarity scores: 30 pairs (Miller and Charles, 1991), 65 pairs (Rubenstein and Goodenough, 1965) and 353 pairs<sup>3</sup> (Finkelstein et al., 2001). We assume that all terms are nouns, so that we can have a fair comparison of the two Thesauri with *WordNet*. We measure the correlation with Pearson’s Correlation Coefficient.

<sup>3</sup><http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/wordsim353.html>

Year	Miller & Charles	Rubenstein & Goodenough	Finkelstein et. al
Index words and phrase			
1911	0.7846	0.7313	0.3449
1987	0.7984	0.7865	0.4214
Index phrase only			
1911	0.7090	0.7168	0.3373
1987	0.7471	0.7777	0.3924

Table 4: Pearson’s coefficient values when not breaking / breaking phrases up.

A preliminary experiment set out to determine whether there is any advantage to indexing the words in a phrase separately, for example, whether the phrase “change of direction” should be indexed only as a whole, or as all of “change”, “of”, “direction” and “change of direction”. The outcome of this experiment appears in Table 4. There is a clear improvement: breaking phrases up gives superior results on all three data sets, for both versions of *Roget’s*. In the remaining experiments, we have each word in a phrase indexed.

We compare the results for the 1911 and 1987 *Roget’s* Thesauri with a variety of *WordNet*-based semantic relatedness measures – see Table 5. We consider 10 measures, noted in the table as J&C (Jiang and Conrath, 1997), Resnik (Resnik, 1995), Lin (Lin, 1998), W&P (Wu and Palmer, 1994), L&C (Leacock and Chodorow, 1998), H&SO (Hirst and St-Onge, 1998), Path (counts edges between synsets), Lesk (Banerjee and Pedersen, 2002), and finally Vector and Vector Pair (Patwardhan, 2003). The latter two work with large vectors of co-occurring terms from a corpus, so *WordNet* is only part of the system. We used Pedersen’s Semantic Distance software package (Pedersen et al., 2004).

The results suggest that neither version of *Roget’s* is best for these data sets. In fact, the Vector method is superior on all three sets, and the Lesk algorithm performs very closely to *Roget’s* 1987. Even on the largest set (Finkelstein et al., 2001), however, the differences between *Roget’s* Thesaurus and the Vector method are not statistically significant at the  $p < 0.05$  level for either thesaurus on a two-tailed test<sup>4</sup>. The difference between the 1911 Thesaurus and Vector *would* be statistically signifi-

<sup>4</sup><http://faculty.vassar.edu/lowry/rdiff.html>

Method	Miller & Charles	Rubenstein & Goodenough	Finkelstein et. al
1911	0.7846	0.7313	0.3449
1987	0.7984	0.7865	0.4214
J&C	0.4735	0.5755	0.2273
Resnik	0.8060	0.8224	0.3531
Lin	0.7388	0.7264	0.2932
W&P	0.7641	0.7973	0.2676
L&C	0.7792	0.8387	0.3094
H&SO	0.6668	0.7258	0.3548
Path	0.7550	0.7842	0.3744
Lesk	0.7954	0.7780	0.4220
Vector	0.8645	0.7929	0.4621
Vct Pair	0.5101	0.5810	0.3722

Table 5: Pearson’s coefficient values for three data sets on a variety of relatedness functions.

cant at  $p < 0.07$ .

On the (Miller and Charles, 1991) and (Rubenstein and Goodenough, 1965) data sets the best system did not show a statistically significant improvement over the 1911 or 1987 *Roget’s* Thesauri, even at  $p < 0.1$  for a two-tailed test. These data sets are too small for a meaningful comparison of systems with close correlation scores.

### 3.2 Synonym identification

In this problem we take a term  $q$  and we seek the correct synonym  $s$  from a set  $C$ . There are two steps. We used the system from (Jarmasz and Szpakowicz, 2004) for identifying synonyms with *Roget’s*. First we find a set of terms  $B \subseteq C$  with the maximum relatedness between  $q$  and each term  $x \in C$ :

$$B = \{x \mid \operatorname{argmax}_{x \in C} \operatorname{semDist}(x, q)\}$$

Next, we take the set of terms  $A \subseteq B$  where each  $a \in A$  has the maximum number of shortest paths between  $a$  and  $q$ .

$$A = \{x \mid \operatorname{argmax}_{x \in B} \operatorname{numberShortestPaths}(x, q)\}$$

If  $s \in A$  and  $|A| = 1$ , the correct synonym has been selected. Often the sets  $A$  and  $B$  will contain just one item. If  $s \in A$  and  $|A| > 1$ , there is a tie. If  $s \notin A$  then the selected synonyms are incorrect. If a multi-word phrase  $c \in C$  of length  $n$  is not found,

ESL						
Method	Yes	Tie	No	QNF	ANF	ONF
1911	27	3	20	0	3	3
1987	36	6	8	0	0	1
J&C	30	4	16	4	4	10
Resnik	26	6	18	4	4	10
Lin	31	5	14	4	4	10
W&P	31	6	13	4	4	10
L&C	29	11	10	4	4	10
H&SO	34	4	12	0	0	0
Path	30	11	9	4	4	10
Lesk	38	0	12	0	0	0
Vector	39	0	11	0	0	0
VctPair	40	0	10	0	0	0
TOEFL						
1911	52	3	25	10	5	25
1987	59	7	14	4	4	17
J&C	34	37	9	33	31	90
Resnik	37	37	6	33	31	90
Lin	33	41	6	33	31	90
W&P	39	36	5	33	31	90
L&C	38	36	6	33	31	90
H&SO	60	16	4	1	0	1
Path	38	36	6	33	31	90
Lesk	70	1	9	1	0	1
Vector	69	1	10	1	0	1
VctPair	65	2	13	1	0	1
RDWP						
1911	157	13	130	57	13	76
1987	198	17	85	22	5	17
J&C	100	146	54	62	58	150
Resnik	114	114	72	62	58	150
Lin	94	160	46	62	58	150
W&P	147	87	66	62	58	150
L&C	149	93	58	62	58	150
H&SO	170	82	48	4	6	5
Path	148	96	56	62	58	150
Lesk	220	7	73	4	6	5
Vector	216	7	73	4	6	5
VctPair	187	10	103	4	6	5

Table 6: Synonym selection experiments.

it is replaced by each of its words  $c_1, c_2, \dots, c_n$ , and each of these words is considered in turn. The  $c_i$  that is closest to  $q$  is chosen to represent  $c$ . When searching for a word in *Roget's* or *WordNet*, we look for all forms of the word.

The results of these experiments appear in Table 6. “Yes” indicates correct answers, “No” – incorrect answers, and “Tie” is for ties. QNF stands for “Question word Not Found”, ANF for “Answer word Not Found” and ONF for “Other word Not Found”. We used three data sets for this application: 80 questions taken from the Test of English as a Foreign Language (TOEFL) (Landauer and Dumais, 1997), 50 questions – from the English as a Second Language test (ESL) (Turney, 2001) and 300 questions – from the Reader’s Digest Word Power Game (RDWP) (Lewis, 2000 and 2001).

Lesk and the Vector-based systems perform better than all others, including *Roget's* 1911 and 1987. Even so, both versions of *Roget's* Thesaurus performed well, and were never worse than the worst *WordNet* systems. In fact, six of the ten *WordNet*-based methods are consistently worse than the 1911 Thesaurus. Since the two Vector-based systems make use of additional data beyond *WordNet*, Lesk is the only completely *WordNet*-based system to outperform *Roget's* 1987. One advantage of *Roget's* Thesaurus is that both versions generally have fewer missing terms than *WordNet*, though Lesk, Hirst & St-Onge and the two vector based methods had fewer missing terms than *Roget's*. This may be because the other *WordNet* methods will only work for nouns and verbs.

### 3.3 Sentence relatedness

Our final experiment concerns sentence relatedness. We worked with a data set from (Li et al., 2006)<sup>5</sup>. They took a subset of the term pairs from (Rubenstein and Goodenough, 1965) and chose sentences to represent these terms; the sentences are definitions from the Collins Cobuild dictionary (Sinclair, 2001). Thirty people were then asked to assign relatedness scores to these sentences, and the average of these similarities was taken for each sentence.

Other methods of determining sentence semantic relatedness expand term relatedness functions to

<sup>5</sup><http://www.docm.mmu.ac.uk/STAFF/D.McLean/SentenceResults.htm>



create a sentence relatedness function (Islam and Inkpen, 2007; Mihalcea et al., 2006). We propose to approach the task by exploiting in other ways the commonalities in the structure of *Roget's* Thesaurus and of *WordNet*. We use the OpenNLP toolkit<sup>6</sup> for segmentation and part-of-speech tagging.

We use a method of sentence representation that involves mapping the sentence into weighted concepts in either *Roget's* or *WordNet*. We mean a concept in *Roget's* to be either a Class, Section, ..., Semicolon Group, while a concept in *WordNet* is any synset. Essentially a concept is a grouping of words from either resource. Concepts are weighted by two criteria. The first is how frequently words from the sentence appear in these concepts. The second is the depth (or specificity) of the concept itself.

### 3.3.1 Weighting based on word frequency

Each word and punctuation mark  $w$  in a sentence is given a score of 1. (Naturally, only open-category words will be found in the thesaurus.) If  $w$  has  $n$  word senses  $w_1, \dots, w_n$ , each sense gets a score of  $1/n$ , so that  $1/n$  is added to each concept in the *Roget's* hierarchy (semicolon group, paragraph, ..., class) or *WordNet* hierarchy that contains  $w_i$ . We weight concepts in this way simply because, unable to determine which sense is correct, we assume that all senses are equally probable. Each concept in *Roget's* Thesaurus and *WordNet* gets the sum of the scores of the concepts below it in its hierarchy.

We will define the scores recursively for a concept  $c$  in a sentence  $s$  and sub-concepts  $c_i$ . For example, in *Roget's* if the concept  $c$  were a Class, then each  $c_i$  would be a Section. Likewise, in *WordNet* if  $c$  were a synset, then each  $c_i$  would be a hyponym synset of  $c$ . Obviously if  $c$  is a word sense  $w_i$  (a word in either a synset or a Semicolon Group), then there can be no sub-concepts  $c_i$ . When  $c = w_i$ , the score for  $c$  is the sum of all occurrences of the word  $w$  in sentence  $s$  divided by the number of senses of the word  $w$ .

$$\text{score}(c, s) = \begin{cases} \frac{\text{instancesOf}(w, s)}{\text{sensesOf}(w)} & \text{if } c = w_i \\ \sum_{c_i \in c} \text{score}(c_i, s) & \text{otherwise} \end{cases}$$

See Table 7 for an example of how this sentence representation works. The sentence “A gem is a jewel or stone that is used in jewellery.” is represented using the 1911 *Roget's*. A concept is identi-

<sup>6</sup><http://opennlp.sourceforge.net>

fied by a name and a series of up to 9 numbers that indicate where in the thesaurus it appears. The first number represents the Class, the second the Section, ..., the ninth the word. We only show concepts with weights greater than 1.0. Words not in the thesaurus keep a weight of 1.0, but this weight will not increase the weight of any concepts in *Roget's* or *WordNet*. Apart from the function words “or”, “in”, “that” and “a” and the period, only the word “jewellery” had a weight above 1.0. The categories labelled 6, 6.2 and 6.2.2 are the only ancestors of the word “use” that ended up with the weights above 1.0. The words “gem”, “is”, “jewel”, “stone” and “used” all contributed weight to the categories shown in Table 7, and to some categories with weights lower than 1.0, but no sense of the words themselves had a weight greater than 1.0.

It is worth noting that this method only relies on the hierarchies in *Roget's* and *WordNet*. We do not take advantage of other *WordNet* relations such as hyponymy, nor do we use any cross-reference links that exist in *Roget's* Thesaurus. Including such relations might improve our sentence relatedness system, but that has been left for future work.

### 3.3.2 Weighting based on specificity

To determine sentence relatedness, one could, for example, flatten the structures like those in Table 7 into vectors and measure their closeness by some vector distance function such as cosine similarity. There is a problem with this, though. A concept inherits the weights of all its sub-concepts, so the concepts that appear closer to the root of the tree will far outweigh others. Some sort of weighting function should be used to re-adjust the weights of particular concepts. Were this an Information Retrieval task, weighting schemes such as *tf.idf* for each concept could apply, but for sentence relatedness we propose an *ad hoc* weighting scheme based on assumptions about which concepts are most important to sentence representation. This weighting scheme is the second element of our sentence relatedness function.

We weight a concept in *Roget's* and in *WordNet* by how many words in a sentence give weight to it. We need to re-weight it based on how specific it is. Clearly, concepts near the leaves of the hierarchy are more specific than those close to the root of the hierarchy. We define specificity as the distance in levels between a given word and each concept found above

Identifier	Concept	Weight
6	Words Relating to the Voluntary Powers - Individual Volition	2.125169028274
6.2	Prospective Volition	1.504066255252
6.2.2	Subservience to Ends	1.128154077172
8	Words Relating to the Sentiment and Moral Powers	3.13220884041
8.2	Personal Affections	1.861744448402
8.2.2	Discriminative Affections	1.636503978149
8.2.2.2	Ornament/Jewelry/Blemish [Head Group]	1.452380952380
8.2.2.2.886	Jewelry [Head]	1.452380952380
8.2.2.2.886.1	Jewelry [Noun]	1.452380952380
8.2.2.2.886.1.1	jewel [Paragraph]	1.452380952380
8.2.2.2.886.1.1.1	jewel [Semicolon Group]	1.166666666666
8.2.2.2.886.1.1.1.3	jewellery [Word Sense]	1.0
or	-	1.0
in	-	1.0
that	-	1.0
a	-	2.0
.	-	1.0

Table 7: “A gem is a jewel or stone that is used in jewellery.” as represented using *Roget’s* 1911.

it in the hierarchy. In *Roget’s* Thesaurus there are exactly 9 levels from the term to the class. In *WordNet* there will be as many levels as a word has ancestors up the hypernymy chain. In *Roget’s*, a term has specificity 1, a Semicolon Group 2, a Paragraph 3, ..., a Class 9. In *WordNet*, the specificity of a word is 1, its synset – 2, the synset’s hypernym – 3, its hypernym – 4, and so on. Words not found in the Thesaurus or in *WordNet* get specificity 1.

We seek a function that, given  $s$ , assigns to all concepts of specificity  $s$  a weight progressively larger than to their neighbours. The weights in this function should be assigned based on specificity, so that all concepts of the same specificity receive the same score. Weights will differ depending on a combination of specificity and how frequently words that signal the concepts appear in a sentence. The weight of concepts with specificity  $s$  should be the highest, of those with specificity  $s \pm 1$  – lower, of those with specificity  $s \pm 2$  lower still, and so on. In order to achieve this effect, we weight the concepts using a normal distribution, where the mean is  $s$ :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{(x-s)^2}{2\sigma^2}\right)}$$

Since the Head is often considered the main category in *Roget’s*, we expect a specificity of 5 to be best, but we decided to test the values 1 through 9 as a possible setting for specificity. We do not claim that this weighting scheme is optimal; other weighting schemes might do better. For the purpose of

comparing the 1911 and 1987 Thesauri and *WordNet*, however, this method appears sufficient.

With this weighting scheme, we determine the distance between two sentences using cosine similarity:

$$\text{cosSim}(A, B) = \frac{\sum a_i * b_i}{\sqrt{\sum a_i^2} * \sqrt{\sum b_i^2}}$$

For this problem we used the MIT Java *WordNet* Interface version 1.1.1<sup>7</sup>.

### 3.3.3 Sentence similarity results

We used this method of representation for *Roget’s* of 1911 and of 1987, as well as for *WordNet* 3.0 – see Figure 1. For comparison, we also implemented a baseline method that we refer to as Simple: we built vectors out of words and their count.

It can be seen in Figure 1 that each system is superior for at least one of the nine specificities. The Simple method is best at a specificity of 1, 8 and 9, *Roget’s* Thesaurus 1911 is best at 6, *Roget’s* Thesaurus 1987 is best at 4, 5 and 7, and *WordNet* is best at 2 and 3. The systems based on *Roget’s* and *WordNet* more or less followed a bell-shaped curve, with the curves of the 1911 and 1987 Thesauri following each other fairly closely and peaking close together. *WordNet* clearly peaked first and then fell the farthest.

<sup>7</sup><http://www.mit.edu/~markaf/projects/wordnet/>

The best correlation result for the 1987 *Roget's* Thesaurus is 0.8725 when the mean is 4, the POS. The maximum correlation for the 1911 Thesaurus is 0.8367, where the mean is 5, the Head. The maximum for *WordNet* is 0.8506, where the mean is 3, or the first hypernym synset. This suggests that the POS and Head are most important for representing text in *Roget's* Thesaurus, while the first hypernym is most important for representing text using *WordNet*. For the Simple method, we found a more modest correlation of 0.6969.

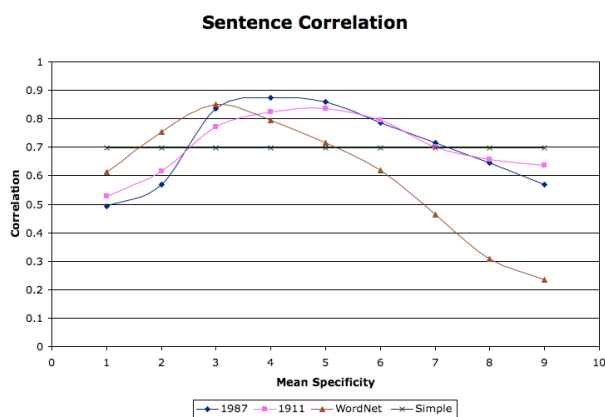


Figure 1: Correlation data for all four systems.

Several other methods have given very good scores on this data set. For the system in (Li et al., 2006), where this data set was first introduced, a correlation of 0.816 with the human annotators was achieved. The mean of all human annotators had a score of 0.825, with a standard deviation of 0.072. In (Islam and Inkpen, 2007), an even better system was proposed, with a correlation of 0.853.

Selecting the mean that gives the best correlation could be considered as training on test data. However, were we simply to have selected a value somewhere in the middle of the graph, as was our original intuition, it would have given an unfair advantage to either version of *Roget's* Thesaurus over *WordNet*. Our system shows good results for both versions of *Roget's* Thesauri and *WordNet*. The 1987 Thesaurus once again performs better than the 1911 version and than *WordNet*. Much like (Miller and Charles, 1991), the data set used here is not large enough to determine if any system's improvement is statistically significant.

## 4 Conclusion and future work

The 1987 version of *Roget's* Thesaurus performed better than the 1911 version on all our tests, but we did not find the differences to be statistically significant. It is particularly interesting that the 1911 Thesaurus performed as well as it did, given that it is almost 100 years old. On problems such as semantic word relatedness, the 1911 Thesaurus performance was fairly close to that of the 1987 Thesaurus, and was comparable to many *WordNet*-based measures. For problems of identifying synonyms both versions of *Roget's* Thesaurus performed relatively well compared to most *WordNet*-based methods.

We have presented a new method of sentence representation that attempts to leverage the structure found in *Roget's* Thesaurus and similar lexical ontologies (among them *WordNet*). We have shown that given this style of text representation both versions of *Roget's* Thesaurus work comparably to *WordNet*. All three perform fairly well compared to the baseline Simple method. Once again, the 1987 version is superior to the 1911 version, but the 1911 version still works quite well.

We hope to investigate further the representation of sentences and other short texts using *Roget's* Thesaurus. These kinds of measurements can help with problems such as identifying relevant sentences for extractive text summarization, or possibly paraphrase identification (Dolan et al., 2004). Another – longer-term – direction of future work could be merging *Roget's* Thesaurus with *WordNet*.

We also plan to study methods of automatically updating the 1911 *Roget's* Thesaurus with modern words. Some work has been done on adding new terms and relations to *WordNet* (Snow et al., 2006) and FACTOTUM (O'Hara and Wiebe, 2003). Similar methods could be used for identifying related terms and assigning them to a correct semicolon group or paragraph.

## Acknowledgments

Our research is supported by the Natural Sciences and Engineering Research Council of Canada and the University of Ottawa. We thank Dr. Diana Inkpen, Anna Kazantseva and Oana Frunza for many useful comments on the paper.

## References

- S. Banerjee and T. Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Proc. CICLing 2002*, pages 136–145.
- P. Cassidy. 2000. An investigation of the semantic relations in the roget’s thesaurus: Preliminary results. In *Proc. CICLing 2000*, pages 181–204.
- B. Dolan, C. Quirk, and C. Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proc. COLING 2004*, pages 350–356, Morristown, NJ.
- C. Fellbaum. 1998. A semantic network of english verbs. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 69–104. MIT Press, Cambridge, MA.
- L. Finkelstein, E. Gabilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2001. Placing search in context: the concept revisited. In *Proc. 10th International Conf. on World Wide Web*, pages 406–414, New York, NY, USA. ACM Press.
- G. Hirst and D. St-Onge. 1998. Lexical chains as representation of context for the detection and correction malapropisms. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 305–322. MIT Press, Cambridge, MA.
- A. Islam and D. Inkpen. 2007. Semantic similarity of short texts. In *Proc. RANLP 2007*, pages 291–297, September.
- M. Jarmasz and S. Szpakowicz. 2003. Not as easy as it seems: Automating the construction of lexical chains using roget’s thesaurus. In *Proc. 16th Canadian Conf. on Artificial Intelligence*, pages 544–549.
- M. Jarmasz and S. Szpakowicz. 2004. Roget’s thesaurus and semantic similarity. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003, Current Issues in Linguistic Theory*, volume 260, pages 111–120. John Benjamins.
- J. Jiang and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. 10th International Conf. on Research on Computational Linguistics*, pages 19–33.
- A. Kennedy and S. Szpakowicz. 2007. Disambiguating hypernym relations for roget’s thesaurus. In *Proc. TSD 2007*, pages 66–75.
- B. Kirkpatrick, editor. 1987. *Roget’s Thesaurus of English Words and Phrases*. Penguin, Harmondsworth, Middlesex, England.
- T. Landauer and S. Dumais. 1997. A solution to Plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240.
- C. Leacock and M. Chodorow. 1998. Combining local context and wordnet sense similarity for word sense disambiguation. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 265–284. MIT Press, Cambridge, MA.
- M. Lewis, editor. 2000 and 2001. *Readers Digest*, 158(932, 934, 935, 936, 937, 938, 939, 940), 159(944, 948). Readers Digest Magazines Canada Limited.
- Y. Li, D. McLean, Z. A. Bandar, J. D. O’Shea, and K. Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1138–1150.
- D. Lin. 1998. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- R. Mihalcea, C. Corley, and C. Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proc. 21st National Conf. on Artificial Intelligence*, pages 775–780. AAAI Press.
- G. A. Miller and W. G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Process*, 6(1):1–28.
- T. P. O’Hara and J. Wiebe. 2003. Classifying functional relations in factotum via wordnet hypernym associations. In *Proc. CICLing 2003*, pages 347–359.
- S. Patwardhan. 2003. Incorporating dictionary and corpus information into a vector measure of semantic relatedness. Master’s thesis, University of Minnesota, Duluth, August.
- T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proc. of the 19th National Conference on Artificial Intelligence.*, pages 1024–1025.
- P. Resnik. 1995. Using information content to evaluate semantic similarity. In *Proc. 14th International Joint Conf. on Artificial Intelligence*, pages 448–453.
- H. Rubenstein and J. B. Goodenough. 1965. Contextual correlates of synonymy. *Communication of the ACM*, 8(10):627–633.
- J. Sinclair. 2001. *Collins Cobuild English Dictionary for Advanced Learners*. Harper Collins Pub.
- R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proc COLING/ACL 2006*, pages 801–808.
- P. Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proc. 12th European Conf. on Machine Learning*, pages 491–502.
- Z. Wu and M. Palmer. 1994. Verb semantics and lexical selection. In *Proc. 32nd Annual Meeting of the ACL*, pages 133–138, New Mexico State University, Las Cruces, New Mexico.

# Unsupervised Translation Induction for Chinese Abbreviations using Monolingual Corpora

Zhifei Li and David Yarowsky

Department of Computer Science and Center for Language and Speech Processing  
Johns Hopkins University, Baltimore, MD 21218, USA  
zhifei.work@gmail.com and yarowsky@cs.jhu.edu

## Abstract

Chinese abbreviations are widely used in modern Chinese texts. Compared with English abbreviations (which are mostly acronyms and truncations), the formation of Chinese abbreviations is much more complex. Due to the richness of Chinese abbreviations, many of them may not appear in available parallel corpora, in which case current machine translation systems simply treat them as unknown words and leave them untranslated. In this paper, we present a novel *unsupervised* method that automatically extracts the relation between a full-form phrase and its abbreviation from monolingual corpora, and induces translation entries for the abbreviation by using its full-form as a bridge. Our method does not require any additional annotated data other than the data that a regular translation system uses. We integrate our method into a state-of-the-art baseline translation system and show that it consistently improves the performance of the baseline system on various NIST MT test sets.

## 1 Introduction

The modern Chinese language is a highly abbreviated one due to the mixed use of ancient single-character words with modern multi-character words and compound words. According to Chang and Lai (2004), approximately 20% of sentences in a typical news article have abbreviated words in them. Abbreviations have become even more popular along with the development of Internet media (e.g., online chat, weblog, newsgroup, and so on). While English words are normally abbreviated by either their

Full-form	Abbreviation	Translation
香港 总督	港督	Hong Kong Governor
安全 理事会	安理会	Security Council

Figure 1: Chinese Abbreviations Examples

first letters (i.e. acronyms) or via truncation, the formation of Chinese abbreviations is much more complex. Figure 1 shows two examples for Chinese abbreviations. Clearly, an abbreviated form of a word can be obtained by selecting *one or more* characters from this word, and the selected characters can be at *any* position in the word. In an extreme case, there are even re-ordering between a full-form phrase and its abbreviation.

While the research in statistical machine translation (SMT) has made significant progress, most SMT systems (Koehn et al., 2003; Chiang, 2007; Galley et al., 2006) rely on parallel corpora to extract translation entries. The richness and complexness of Chinese abbreviations imposes challenges to the SMT systems. In particular, many Chinese abbreviations may not appear in available parallel corpora, in which case current SMT systems treat them as unknown words and leave them untranslated. This affects the translation quality significantly.

To be able to translate a Chinese abbreviation that is unseen in available parallel corpora, one may annotate more parallel data. However, this is very expensive as there are too many possible abbreviations and new abbreviations are constantly created. Another approach is to transform the abbreviation

into its full-form for which the current SMT system knows how to translate. For example, if the baseline system knows that the translation for “香港总督” is “Hong Kong Governor”, and it also knows that “港督” is an abbreviation of “香港总督”, then it can translate “港督” to “Hong Kong Governor”.

Even if an abbreviation has been seen in parallel corpora, it may still be worth to consider its full-form phrase as an *additional* alternative to the abbreviation since abbreviated words are normally semantically ambiguous, while its full-form contains more context information that helps the MT system choose a right translation for the abbreviation.

Conceptually, the approach of translating an abbreviation by using its full-form as a bridge involves four components: identifying abbreviations, learning their full-forms, inducing their translations, and integrating the abbreviation translations into the baseline SMT system. None of these components is trivial to realize. For example, for the first two components, we may need manually annotated data that tags an abbreviation with its full-form. We also need to make sure that the baseline system has at least one valid translation for the full-form phrase. On the other hand, integrating an additional component into a baseline SMT system is notoriously tricky as evident in the research on integrating word sense disambiguation (WSD) into SMT systems: different ways of integration lead to conflicting conclusions on whether WSD helps MT performance (Chan et al., 2007; Carpuat and Wu, 2007).

In this paper, we present an unsupervised approach to translate Chinese abbreviations. Our approach exploits the *data co-occurrence* phenomena and does not require any additional annotated data except the parallel and monolingual corpora that the baseline SMT system uses. Moreover, our approach integrates the abbreviation translation component into the baseline system in a natural way, and thus is able to make use of the minimum-error-rate training (Och, 2003) to automatically adjust the model parameters to reflect the change of the integrated system over the baseline system. We carry out experiments on a state-of-the-art SMT system, i.e., Moses (Koehn et al., 2007), and show that the abbreviation translations consistently improve the translation performance (in terms of BLEU (Papineni et al., 2002)) on various NIST MT test sets.

## 2 Background: Chinese Abbreviations

In general, Chinese abbreviations are formed based on three major methods: *reduction*, *elimination* and *generalization* (Lee, 2005; Yin, 1999). Table 1 presents examples for each category.

Among the three methods, *reduction* is the most popular one, which generates an abbreviation by selecting one or more characters from each of the words in the full-form phrase. The selected characters can be at any position of the word. Table 1 presents examples to illustrate how characters at different positions are selected to generate abbreviations. While the abbreviations mostly originate from noun phrases (in particular, named entities), other general phrases are also abbreviatable. For example, the second example “Save Energy” is a verb phrase. In an extreme case, reordering may happen between an abbreviation and its full-form phrase. For example, for the seventh example in Table 1, a monotone abbreviation should be “一核厂”, however, “核一厂” is a more popular ordering in Chinese texts.

In *elimination*, one or more words of the original full-form phrase are eliminated and the rest parts remain as an abbreviation. For example, in the full-form phrase “清华大学”, the word “大学” is eliminated and the remaining word “清华” alone becomes the abbreviation.

In *generalization*, an abbreviation is created by generalizing parallel sub-parts of the full-form phrase. For example, “三防 (three preventions)” in Table 1 is an abbreviation for the phrase “防火、防盗、防交通事故 (fire prevention, theft prevention, and traffic accident prevention)”. The character “防 (prevention)” is common to the three sub-parts of the full-form, so it is being generalized.

## 3 Unsupervised Translation Induction for Chinese Abbreviations

In this section, we describe an *unsupervised* method to induce translation entries for Chinese abbreviations, even when these abbreviations never appear in the Chinese side of the parallel corpora. Our basic idea is to automatically extract the relation between a full-form phrase and its abbreviation (we refer the relation as *full-abbreviation*) from monolingual corpora, and then induce translation entries for the abbreviation by using its full-form phrase as a bridge.

Category	Full-form	Abbreviation	Translation
<b>Reduction</b>	北京 大学	北大	Peking University
	节约 能源	节能	Save Energy
	香港 总督	港督	Hong Kong Governor
	外交 部长	外长	Foreign Minister
	人民 警察	民警	People’s Police
	安全 理事会	安理会	Security Council
	第二 核能 发电厂	核一厂	No.1 Nuclear Energy Power Plant
<b>Elimination</b>	清华 大学	清华	Tsinghua University
<b>Generalization</b>	防火、防盗、防交通事故	三防	Three Preventions

Table 1: Chinese Abbreviation: Categories and Examples

Our approach involves five major steps:

- Step-1: extract a list of English entities from English monolingual corpora;
- Step-2: translate the list into Chinese using a baseline translation system;
- Step-3: extract *full-abbreviation* relations from Chinese monolingual corpora by treating the Chinese translations obtained in Step-2 as full-form phrases;
- Step-4: induce translation entries for Chinese abbreviations by using their full-form phrases as bridges;
- Step-5: augment the baseline system with translation entries obtained in Step-4.

Clearly, the main purpose of Step-1 and -2 is to obtain a list of Chinese entities, which will be treated as full-form phrases in Step-3. One may use a named entity tagger to obtain such a list. However, this relies on the existence of a Chinese named entity tagger with high-precision. Moreover, obtaining a list using a dedicated tagger does not guarantee that the baseline system knows how to translate the list. On the contrary, in our approach, since the Chinese entities are translation outputs for the English entities, it is ensured that the baseline system has translations for these Chinese entities.

Regarding the data resource used, Step-1, -2, and -3 rely on the English monolingual corpora, parallel corpora, and the Chinese monolingual corpora, *respectively*. Clearly, our approach does not require any additional annotated data compared with

the baseline system. Moreover, our approach utilizes both Chinese and English monolingual data to help MT, while most SMT systems utilizes only the English monolingual data to build a language model. This is particularly interesting since we normally have enormous monolingual data, but a small amount of parallel data. For example, in the translation task between Chinese and English, both the Chinese and English Gigaword have billions of words, but the parallel data has only about 30 million words.

Step-4 and -5 are natural ways to integrate the abbreviation translation component with the baseline translation system. This is critical to make the abbreviation translation get performance gains over the baseline system as will be clear later.

In the remainder of this section, we will present a specific instantiation for each step.

### 3.1 English Entity Extraction from English Monolingual Corpora

Though one can exploit a sophisticated named-entity tagger to extract English entities, in this paper we identify English entities based on the capitalization information. Specifically, to be considered as an entity, a continuous span of English words must satisfy the following conditions:

- all words must start from a capital letter except for function words “of”, “the”, and “and”;
- each function word can appear only once;
- the number of words in the span must be smaller than a threshold (e.g., 10);
- the occurrence count of this span must be greater than a threshold (e.g., 1).

### 3.2 English Entity Translation

For the Chinese-English language pair, most MT research is on translation from Chinese to English, but here we need the reverse direction. However, since most of statistical translation models (Koehn et al., 2003; Chiang, 2007; Galley et al., 2006) are *symmetrical*, it is relatively easy to train a translation system to translate from English to Chinese, except that we need to train a Chinese language model from the Chinese monolingual data.

It is worth pointing out that the baseline system may not be able to translate all the English entities. This is because the entities are extracted from the English monolingual corpora, which has a much larger vocabulary than the English side of the parallel corpora. Therefore, we should remove all the Chinese translations that contain any untranslated English words before proceeding to the next step. Moreover, it is desirable to generate an n-best list instead of a 1-best translation for the English entity.

### 3.3 Full-abbreviation Relation Extraction from Chinese Monolingual Corpora

We treat the Chinese entities obtained in Section 3.2 as full-form phrases. To identify their abbreviations, one can employ an HMM model (Chang and Teng, 2006). Here we propose a much simpler approach, which is based on the *data co-occurrence* intuition.

#### 3.3.1 Data Co-occurrence

In a monolingual corpus, relevant words tend to appear together (i.e., co-occurrence). For example, *Bill Gates* tends to appear together with *Microsoft*. The co-occurrence may imply a relationship (e.g., *Bill Gates* is the founder of *Microsoft*). By inspection of the Chinese text, we found that the *data co-occurrence* phenomena also applies to the *full-*

Title	都灵冬奥会开幕式将激情上演
Text	新华社都灵2月9日电(记者丁莹 阎涛)第20届冬季奥运会的开幕式将于当地时间10日晚8点在都灵奥林匹克体育场正式揭开神秘的面纱。

Table 2: Data Co-occurrence Example for the *Full-abbreviation Relation* (冬季奥运会, 冬奥会) meaning “winter olympics”

*abbreviation* relation. Table 2 shows an example, where the abbreviation “冬奥会” appears in the title while its full-form “冬季奥运会” appears in the text of the same document. In general, the occurrence distance between an abbreviation and its full-form varies. For example, they may appear in the same sentence, or in the neighborhood sentences.

#### 3.3.2 Full-abbreviation Relation Extraction Algorithm

By exploiting the *data co-occurrence* phenomena, we identify possible abbreviations for full-form phrases. Figure 2 presents the pseudocode of the *full-abbreviation* relation extraction algorithm.

#### Relation-Extraction(*Corpus*, *Full-list*)

```

1 contexts ← NIL
2 for i ← 1 to length[Corpus]
3   sent1 ← Corpus[i]
4   contexts ← UPDATE(contexts, Corpus, i)
5   for full in sent1
6     if full in Full-list
7       for sent2 in contexts
8         for abbr in sent2
9           if RL(full, abbr) = TRUE
10            Count[abbr, full]++
11 return Count

```

Figure 2: *Full-abbreviation* Relation Extraction

Given a monolingual corpus and a list of full-form phrases (i.e., *Full-list*, which is obtained in Section 3.2), the algorithm returns a *Count* that contains *full-abbreviation* relations and their occurrence counts. Specifically, the algorithm linearly scans over the whole corpus as indicated by line 1. Along the linear scan, the algorithm maintains *contexts* of the current sentence (i.e., *sent1*), and the *contexts* remember the sentences from where the algorithm identifies possible abbreviations. In our implementation, the *contexts* include current sentence, the title of current document, and previous and next sentence in the document. Then, for each ngram (i.e., *full*) of the current sentence (i.e., *sent1*) and for each ngram (i.e., *abbr*) of a context sentence (i.e., *sent2*), the algorithm calls a function *RL*, which decides whether the *full-abbreviation* relation holds between *full* and *abbr*. If *RL* returns TRUE, the count table



(i.e., *Count*) is incremented by one for this relation. Note that the filtering through the full-form phrases list (i.e., *Full-list*) as shown in line 6 is the key to make the algorithm efficient enough to run through large-size monolingual corpora.

In function RL, we run a simple alignment algorithm that links the characters in *abbr* with the words in *full*. In the alignment, we assume that there is no reordering between *full* and *abbr*. To be considered as a valid *full-abbreviation* relation, *full* and *abbr* must satisfy the following conditions:

- *abbr* must be shorter than *full* by a relative threshold (e.g., 1.2);
- each character in *abbr* must be aligned to *full*;
- each word in *full* must have at least one character aligned to *abbr*;
- *abbr* must *not* be a continuous sub-part of *full*;

Clearly, due to the above conditions, our approach may not be able to handle all possible abbreviations (e.g., the abbreviations formed by the *generalization* method described in Section 2). One can modify the conditions and the alignment algorithm to handle more complex *full-abbreviation* relations.

With the count table *Count*, we can calculate the relative frequency and get the following probability,

$$P(full|abbr) = \frac{Count[abbr, full]}{\sum Count[abbr, *]} \quad (1)$$

### 3.4 Translation Induction for Chinese Abbreviations

Given a Chinese abbreviation and its full-form, we induce English translation entries for the abbreviation by using the full-form as a bridge. Specifically, we first generate n-best translations for each full-form Chinese phrase using the baseline system.<sup>1</sup> We then post-process the translation outputs such that they have the same format (i.e., containing the same set of model features) as a regular phrase entry in

<sup>1</sup>In our method, it is guaranteed that each Chinese full-form phrase will have at least one English translation, i.e., the English entity that has been used to produce this full-form phrase. However, it does not mean that this English entity is the *best* translation that the baseline system has for the Chinese full-form phrase. This is mainly due to the *asymmetry* introduced by the different LMs in different translation directions.

the baseline phrase table. Once we get the translation entries for the full-form, we can replace the full-form Chinese with its abbreviation to generate translation entries for the abbreviation. Moreover, to deal with the case that an abbreviation may have several candidate full-form phrases, we normalize the feature values using the following equation,

$$\Phi_j(e, abbr) = \Phi_j(e, full) \times P(full|abbr) \quad (2)$$

where *e* is an English translation, and  $\Phi_j$  is the *j*-th model feature indexed as in the baseline system.

### 3.5 Integration with Baseline Translation System

Since the obtained translation entries for abbreviations have the same format as the regular translation entries in the baseline phrase table, it is relatively easy to add them into the baseline phrase table. Specifically, if a translation entry (signed by its Chinese and English strings) to be added is not in the baseline phrase table, we simply add the entry into the baseline table. On the other hand, if the entry is already in the baseline phrase table, then we *merge* the entries by *enforcing* the translation probability as we obtain the same translation entry from two different knowledge sources (one is from parallel corpora and the other one is from the Chinese monolingual corpora).

Once we obtain the augmented phrase table, we should run the minimum-error-rate training (Och, 2003) with the augmented phrase table such that the model parameters are properly adjusted. As will be shown in the experimental results, this is critical to obtain performance gain over the baseline system.

## 4 Experimental Results

### 4.1 Corpora

We compile a parallel dataset which consists of various corpora distributed by the Linguistic Data Consortium (LDC) for NIST MT evaluation. The parallel dataset has about 1M sentence pairs, and about 28M words. The monolingual data we use includes the English Gigaword V2 (LDC2005T12) and the Chinese Gigaword V2 (LDC2005T14).

### 4.2 Baseline System Training

Using the toolkit Moses (Koehn et al., 2007), we built a phrase-based baseline system by following

the standard procedure: running GIZA++ (Och and Ney, 2000) in both directions, applying refinement rules to obtain a many-to-many word alignment, and then extracting and scoring phrases using heuristics (Och and Ney, 2004). The baseline system has eight feature functions (see Table 8). The feature functions are combined under a log-linear framework, and the weights are tuned by the minimum-error-rate training (Och, 2003) using BLEU (Papineni et al., 2002) as the optimization metric.

To handle different directions of translation between Chinese and English, we built two trigram language models with modified Kneser-Ney smoothing (Chen and Goodman, 1998) using the SRILM toolkit (Stolcke, 2002).

### 4.3 Statistics on Intermediate Steps

As described in Section 3, our approach involves five major steps. Table 3 reports the statistics for each intermediate step. While about 5M English entities are extracted and 2-best Chinese translations are generated for each English entity, we get only 4.7M Chinese entities. This is because many of the English entities are untranslatable by the baseline system. The number of *full-abbreviation* relations<sup>2</sup> extracted from the Chinese monolingual corpora is 51K. For each full-form phrase we generate 5-best English translations, however only 210k ( $<51K \times 5$ ) translation entries are obtained. This is because the baseline system may have less than 5 unique translations for some of the full-form phrases. Lastly, the number of translation entries added due to abbreviations is very small compared with the total number of translation entries (i.e., 50M).

Measure	Value
number of English entities	5M
number of Chinese entities	4.7M
number of <i>full-abbreviation</i> relations	51K
number of translation entries added	210K
total number of translation entries	50M

Table 3: Statistics on Intermediate Steps

<sup>2</sup>Note that many of the “abbreviations” extracted by our algorithm are not true abbreviations in the linguistic sense, instead they are just continuous-span of words. This is analogous to the concept of “phrase” in phrase-based MT.

### 4.4 Precision on *Full-abbreviation* Relations

Table 4 reports the precision on the extracted *full-abbreviation* relations. We classify the relations into several classes based on their occurrence counts. In the second column, we list the fraction of the relations in the given class among all the relations we have extracted (i.e., 51K relations). For each class, we randomly select 100 relations, manually tag them as correct or wrong, and then calculate the precision. Intuitively, a class that has a higher occurrence count should have a higher precision, and this is generally true as shown in the fourth column of Table 4. In comparison, Chang and Teng (2006) reports a precision of 50% over relations between *single-word* full-forms and *single-character* abbreviations. One can imagine a much lower precision on general relations (e.g., the relations between *multi-word* full-forms and *multi-character* abbreviations) that we consider here. Clearly, our results are very competitive<sup>3</sup>.

Count	Fraction (%)	Precision (%)	
		Baseline	Ours
(0, 1]	35.2	8.9	42.6
(1, 5]	33.8	7.8	54.4
(5, 10]	10.7	8.9	60.0
(10, 100]	16.5	7.6	55.9
(100, +∞)	3.8	12.1	59.9
<b>Average Precision (%)</b>		8.4	<b>51.3</b>

Table 4: *Full-abbreviation* Relation Extraction Precision

To further show the advantage of our relation extraction algorithm (see Section 3.3), in the third column of Table 4 we report the results on a simple baseline. To create the baseline, we make use of the *dominant* abbreviation patterns shown in Table 5, which have been reported in Chang and Lai (2004). The abbreviation pattern is represented using the format “(*bit pattern*|*length*)” where the *bit pattern* encodes the information about how an abbreviated form is obtained from its original full-form *word*, and the *length* represents the number of characters in the full-form *word*. In the *bit pattern*, a “1” indicates that the character at the corresponding position of the full-form word is kept in the abbreviation, while a “0” means the character is deleted. Now we dis-

<sup>3</sup>However, it is not a strict comparison because the dataset is different and the *recall* may also be different.

Pattern	Fraction (%)	Example
(1 1)	100	(中, 中)
(10 2)	87	(亚洲, 亚)
(101 3)	44	(理事会, 理会)
(1010 4)	56	(公民投票, 公投)

Table 5: Dominant Abbreviation Patterns reported in Chang and Lai (2004)

cuss how to create the baseline. For each full-form phrase in the randomly selected relations, we generate a baseline hypothesis (i.e., abbreviation) as follows. We first generate an abbreviated form for each *word* in the full-form *phrase* by using the dominant abbreviation pattern, and then concatenate these abbreviated words to form a baseline abbreviation for the full-form *phrase*. As shown in Table 4, the baseline performs significantly worse than our relation extraction algorithm. Compared with the baseline, our relation extraction algorithm allows arbitrary abbreviation patterns as long as they satisfy the alignment constraints. Moreover, our algorithm exploits the *data co-occurrence* phenomena to generate and rank hypothesis (i.e., abbreviation). The above two reasons explain the large performance gain.

It is interesting to examine the statistics on abbreviation patterns over the relations automatically extracted by our algorithm. Table 6 reports the statistics. We obtain the statistics on the relations that are manually tagged as correct before, and there are in total 263 unique *words* in the corresponding full-form *phrases*. Note that the results here are highly biased to our relation extraction algorithm (see Section 3.3). For the statistics on *manually* collected examples, please refer to Chang and Lai (2004).

## 4.5 Results on Translation Performance

### 4.5.1 Precision on Translations of Chinese Full-form Phrases

For the relations manually tagged as correct in Section 4.4, we manually look at the top-5 translations for the full-form phrases. If the top-5 translations contain at least one correct translation, we tag it as correct, otherwise as wrong. We get a precision of 97.5%. This precision is extremely high because the BLEU score (precision with brevity penalty) that one obtains for a Chinese sentence is normally between 30% to 50%. Two reasons explain such a high

Pattern	Fraction (%)	Example
(1 1)	100	(中, 中)
(10 2)	74.3	(亚洲, 亚)
(01 2)	7.6	(北京, 京)
(11 2)	18.1	(早餐, 早餐)
(100 3)	58.5	(伊拉克, 伊)
(010 3)	3.1	(行政院, 政)
(001 3)	4.6	(研究所, 所)
(110 3)	13.8	(奥运会, 奥运)
(101 3)	3.1	(理事会, 理会)
(111 3)	16.9	(科学家, 科学家)

Table 6: Statistics on Abbreviation Patterns

precision. Firstly, the full-form phrase is short compared with a regular Chinese sentence, and thus it is easier to translate. Secondly, the full-form phrase itself contains enough context information that helps the system choose a right translation for it. In fact, this shows the importance of considering the full-form phrase as an *additional* alternative to the abbreviation even if the baseline system already has translation entries for the abbreviation.

### 4.5.2 BLEU on NIST MT Test Sets

We use MT02 as the development set<sup>4</sup> for minimum error rate training (MERT) (Och, 2003). The MT performance is measured by lower-case 4-gram BLEU (Papineni et al., 2002). Table 7 reports the results on various NIST MT test sets. As shown in the table, our Abbreviation Augmented MT (AAMT) systems perform consistently better than the baseline system (described in Section 4.2).

Task	Baseline	AAMT	
		No MERT	With MERT
MT02	29.87	29.96	<b>30.46</b>
MT03	29.03	29.23	<b>29.71</b>
MT04	29.05	29.88	<b>30.55</b>
<b>Average Gain</b>		+0.52	<b>+1.18</b>

Table 7: MT Performance measured by BLEU Score

As clear in Table 7, it is important to re-run MERT (on MT02 only) with the augmented phrase table in order to get performance gains. Table 8 reports

<sup>4</sup>On the dev set, about 20K (among 210K) abbreviation translation entries are matched in the Chinese side.

the MERT weights with different phrase tables. One may notice the change of the weight in *word penalty* feature. This is very intuitive in order to prevent the hypothesis being too long due to the expansion of the abbreviations into their full-forms.

Feature	Baseline	AAMT
language model	0.137	0.133
phrase translation	0.066	0.023
lexical translation	0.061	0.078
reverse phrase translation	0.059	0.103
reverse lexical translation	0.112	0.090
phrase penalty	-0.150	-0.162
<b>word penalty</b>	<b>-0.327</b>	<b>-0.356</b>
distortion model	0.089	0.055

Table 8: Weights obtained by MERT

## 5 Related Work

Though automatically extracting the relations between full-form Chinese phrases and their abbreviations is an interesting and important task for many natural language processing applications (e.g., machine translation, question answering, information retrieval, and so on), not much work is available in the literature. Recently, Chang and Lai (2004), Chang and Teng (2006), and Lee (2005) have investigated this task. Specifically, Chang and Lai (2004) describes a hidden markov model (HMM) to model the relationship between a full-form phrase and its abbreviation, by treating the abbreviation as the *observation* and the full-form words as *states* in the model. Using a set of manually-created *full-abbreviation* relations as training data, they report experimental results on a *recognition* task (i.e., given an abbreviation, the task is to obtain its full-form, or the vice versa). Clearly, their method is *supervised* because it requires the *full-abbreviation* relations as training data.<sup>5</sup> Chang and Teng (2006) extends the work in Chang and Lai (2004) to automatically extract the relations between full-form phrases and their abbreviations. However, they have only considered relations between single-word phrases and single-character abbreviations. Moreover, the HMM model is computationally-expensive and unable to exploit the *data co-occurrence* phenomena that we

<sup>5</sup>However, the HMM model aligns the characters in the abbreviation to the words in the full-form in an *unsupervised* way.

have exploited efficiently in this paper. Lee (2005) gives a summary about how Chinese abbreviations are formed and presents many examples. Manual rules are created to expand an abbreviation to its full-form, however, no quantitative results are reported.

None of the above work has addressed the Chinese abbreviation issue in the context of a machine translation task, which is the primary goal in this paper. To the best of our knowledge, our work is the first to systematically model Chinese abbreviation expansion to improve machine translation.

The idea of using a bridge (i.e., full-form) to obtain translation entries for unseen words (i.e., abbreviation) is similar to the idea of using paraphrases in MT (see Callison-Burch et al. (2006) and references therein) as both are trying to introduce *generalization* into MT. At last, the goal that we aim to exploit monolingual corpora to help MT is in-spirit similar to the goal of using non-parallel corpora to help MT as aimed in a large amount of work (see Munteanu and Marcu (2006) and references therein).

## 6 Conclusions

In this paper, we present a novel method that automatically extracts relations between full-form phrases and their abbreviations from monolingual corpora, and induces translation entries for these abbreviations by using their full-form as a bridge. Our method is *scalable* enough to handle large amount of monolingual data, and is essentially *unsupervised* as it does not require any additional annotated data than the baseline translation system. Our method exploits the *data co-occurrence* phenomena that is very useful for relation extractions. We integrate our method into a state-of-the-art phrase-based baseline translation system, i.e., Moses (Koehn et al., 2007), and show that the integrated system consistently improves the performance of the baseline system on various NIST machine translation test sets.

## Acknowledgments

We would like to thank Yi Su, Sanjeev Khudanpur, Philip Resnik, Smaranda Muresan, Chris Dyer and the anonymous reviewers for their helpful comments. This work was partially supported by the Defense Advanced Research Projects Agency's GALE program via Contract No HR0011-06-2-0001.

## References

- Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Translation Using Paraphrases. *In Proceedings of NAACL 2006*, pages 17-24.
- Marine Carpuat and Dekai Wu. 2007. Improving Statistical Machine Translation using Word Sense Disambiguation. *In Proceedings of EMNLP 2007*, pages 61-72.
- Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word Sense Disambiguation Improves Statistical Machine Translation. *In Proceedings of ACL 2007*, pages 33-40.
- Jing-Shin Chang and Yu-Tso Lai. 2004. A preliminary study on probabilistic models for Chinese abbreviations. *In Proceedings of the 3rd SIGHAN Workshop on Chinese Language Processing*, pages 9-16.
- Jing-Shin Chang and Wei-Lun Teng. 2006. Mining Atomic Chinese Abbreviation Pairs: A Probabilistic Model for Single Character Word Recovery. *In Proceedings of the 5th SIGHAN Workshop on Chinese Language Processing*, pages 17-24.
- Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.
- David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201-228.
- Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeeffe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. *In Proceedings of COLING/ACL 2006*, pages 961-968.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. *In Proceedings of ACL*, Demonstration Session, pages 177-180.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. *In Proceedings of NAACL 2003*, pages 48-54.
- H.W.D Lee. 2005. A study of automatic expansion of Chinese abbreviations. MA Thesis, The University of Hong Kong.
- Dragos Stefan Munteanu and Daniel Marcu. 2006. Extracting Parallel Sub-Sentential Fragments from Non-Parallel Corpora. *In Proceedings of ACL 2006*, pages 81-88.
- Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. *In Proceedings of ACL 2003*, pages 160-167.
- Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. *In Proceedings of ACL 2000*, pages 440-447.
- Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417-449.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. *In Proceedings of ACL 2002*, pages 311-318.
- Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. *In Proceedings of the International Conference on Spoken Language Processing*, pages 901-904.
- Z.P. Yin. 1999. Methodologies and principles of Chinese abbreviation formation. *In Language Teaching and Study*, 2:73-82.

# Which Are the Best Features for Automatic Verb Classification

**Jianguo Li**

Department of Linguistics  
The Ohio State University  
Columbus Ohio, USA

jianguo@ling.ohio-state.edu

**Chris Brew**

Department of Linguistics  
The Ohio State University  
Columbus Ohio, USA

cbrew@ling.ohio-state.edu

## Abstract

In this work, we develop and evaluate a wide range of feature spaces for deriving Levin-style verb classifications (Levin, 1993). We perform the classification experiments using Bayesian Multinomial Regression (an efficient log-linear modeling framework which we found to outperform SVMs for this task) with the proposed feature spaces. Our experiments suggest that subcategorization frames are not the most effective features for automatic verb classification. A mixture of syntactic information and lexical information works best for this task.

## 1 Introduction

Much research in lexical acquisition of verbs has concentrated on the relation between verbs and their argument frames. Many scholars hypothesize that the behavior of a verb, particularly with respect to the expression of arguments and the assignment of semantic roles is to a large extent driven by deep semantic regularities (Dowty, 1991; Green, 1974; Goldberg, 1995; Levin, 1993). Thus measurements of verb frame patterns can perhaps be used to probe for linguistically relevant aspects of verb meanings. The correspondence between meaning regularities and syntax has been extensively studied in Levin (1993) (hereafter Levin). Levin's verb classes are based on the ability of a verb to occur or not occur in pairs of syntactic frames that are in some sense meaning preserving (*diathesis alternation*). The focus is on verbs for which distribution of syntactic frames is a useful indicator of class membership,

and, correspondingly, on classes which are relevant for such verbs. By using Levin's classification, we obtain a window on some (but not all) of the potentially useful semantic properties of verbs.

Levin's verb classification, like others, helps reduce redundancy in verb descriptions and enables generalizations across semantically similar verbs with respect to their usage. When the information about a verb type is not available or sufficient for us to draw firm conclusions about its usage, the information about the class to which the verb type belongs can compensate for it, addressing the pervasive problem of data sparsity in a wide range of NLP tasks, such as automatic extraction of subcategorization frames (Korhonen, 2002), semantic role labeling (Swier and Stevenson, 2004; Gildea and Jurafsky, 2002), natural language generation for machine translation (Habash et al., 2003), and deriving predominant verb senses from unlabeled data (Lapata and Brew, 2004).

Although there exist several manually-created verb lexicons or ontologies, including Levin's verb taxonomy, VerbNet, and FrameNet, automatic verb classification (AVC) is still necessary for extending existing lexicons (Korhonen and Briscoe, 2004), building and tuning lexical information specific to different domains (Korhonen et al., 2006), and bootstrapping verb lexicons for new languages (Tsang et al., 2002).

AVC helps avoid the expensive hand-coding of such information, but appropriate features must be identified and demonstrated to be effective. In this work, our primary goal is not necessarily to obtain the optimal classification, but rather to investigate

the linguistic conditions which are crucial for lexical semantic classification of verbs. We develop feature sets that combine syntactic and lexical information, which are in principle useful for any Levin-style verb classification. We test the general applicability and scalability of each feature set to the distinctions among 48 verb classes involving 1,300 verbs, which is, to our knowledge, the largest investigation on English verb classification by far. To preview our results, a feature set that combines both syntactic information and lexical information works much better than either of them used alone. In addition, mixed feature sets also show potential for scaling well when dealing with larger number of verbs and verb classes. In contrast, subcategorization frames, at least on their own, are largely ineffective for AVC, despite their evident effectiveness in supporting Levin's initial intuitions.

## 2 Related Work

Earlier work on verb classification has generally adopted one of the two approaches for devising statistical, corpus-based features.

**Subcategorization frame (SCF):** Subcategorization frames are obviously relevant to alternation behaviors. It is therefore unsurprising that much work on verb classification has adopted them as features (Schulte im Walde, 2000; Brew and Schulte im Walde, 2002; Korhonen et al., 2003). However, relying solely on subcategorization frames also leads to the loss of semantic distinctions. Consider the frame NP-V-PP*with*. The semantic interpretation of this frame depends to a large extent on the NP argument selected by the preposition *with*. In (1), the same surface form NP-V-PP*with* corresponds to three different underlying meanings. However, such semantic distinctions are totally lost if lexical information is disregarded.

- (1) a. I ate with *a fork*. [INSTRUMENT]  
 b. I left with *a friend*. [ACCOMPANIMENT]  
 c. I sang with *confidence*. [MANNER]

This deficiency of unlexicalized subcategorization frames leads researchers to make attempts to incorporate lexical information into the feature representation. One possible improvement over subcategorization frames is to enrich them with lexical information. Lexicalized frames are usually obtained

by augmenting each syntactic slot with its head noun (2).

- (2) a. NP(I)-V-PP(*with:fork*)  
 b. NP(I)-V-PP(*with:friend*)  
 c. NP(I)-V-PP(*with:confidence*)

With the potentially improved discriminatory power also comes increased exposure to sparse data problems. Trying to overcome the problem of data sparsity, Schulte im Walde (2000) explores the additional use of selectional preference features by augmenting each syntactic slot with the concept to which its head noun belongs in an ontology (e.g. WordNet). Although the problem of data sparsity is alleviated to certain extent (3), these features do not generally improve classification performance (Schulte im Walde, 2000; Joanis, 2002).

- (3) a. NP(PERSON)-V-PP(*with:ARTIFACT*)  
 b. NP(PERSON)-V-PP(*with:PERSON*)  
 c. NP(PERSON)-V-PP(*with:FEELING*)

**JOANIS07:** Incorporating lexical information directly into subcategorization frames has proved inadequate for AVC. Other methods for combining syntactic information with lexical information have also been attempted (Merlo and Stevenson, 2001; Joanis et al., 2007). These studies use a small collection of features that require some degree of expert linguistic analysis to devise. The deeper linguistic analysis allows their feature set to cover a variety of indicators of verb semantics, beyond that of frame information. Joanis et al. (2007) reports an experiment that involves 15 Levin verb classes. They define a general feature space that is supposed to be applicable to all Levin classes. The features they use fall into four different groups: *syntactic slots*, *slot overlaps*, *tense*, *voice and aspect*, and *animacy of NPs*.

- *Syntactic slots*: They encode the frequency of the syntactic positions (e.g. SUBJECT, OBJECT, PPat). They are considered approximation to subcategorization frames.
- *Slot overlaps*: They are supposed to capture the properties of alternation by identifying if a given noun can occur in different syntactic positions relative to a particular verb. For instance, in the alternation *The ice melted* and

*The sun melted the ice*, *ice* occurs in the subject position in the first sentence but in the object position in the second sentence. An overlap feature records that there is a subject-object alternation for *melt*.

- *Tense, voice and aspect*: Verb meaning and alternations also interact in interesting ways with *tense*, *voice*, and *aspect*. For example, *middle* construction is usually used in present tense (e.g. *The bread cuts easily*).
- *Animacy of NPs*: The animacy of the semantic role corresponding to the head noun in each syntactic slot can also distinguish classes of verbs.

Joanis et al. (2007) demonstrates that the general feature space they devise achieves a rate of error reduction ranging from 48% to 88% over a chance baseline accuracy, across classification tasks of varying difficulty. However, they also show that their general feature space does not generally improve the classification accuracy over subcategorization frames (see table 1).

Experimental Task	All Features	SCF
Average 2-way	83.2	80.4
Average 3-way	69.6	69.4
Average ( $\geq 6$ )-way	61.1	62.8

Table 1: Results from Joanis et al. (2007) (%)

### 3 Integration of Syntactic and Lexical Information

In this study, we explore a wider range of features for AVC, focusing particularly on various ways to mix syntactic with lexical information.

**Dependency relation (DR)**: Our way to overcome data sparsity is to break lexicalized frames into *lexicalized slots* (a.k.a. dependency relations). Dependency relations contain both syntactic and lexical information (4).

- (4)
- SUBJ(*I*), PP(*with:fork*)
  - SUBJ(*I*), PP(*with:friend*)
  - SUBJ(*I*), PP(*with:confidence*)

However, augmenting PP with nouns selected by the preposition (e.g. PP(*with:fork*)) still gives rise

to data sparsity. We therefore decide to break it into two individual dependency relations: PP(*with*), PP(*fork*). Although dependency relations have been widely used in automatic acquisition of lexical information, such as detection of polysemy (Lin, 1998) and WSD (McCarthy et al., 2004), their utility in AVC still remains untested.

**Co-occurrence (CO)**: CO features mostly convey lexical information only and are generally considered not particularly sensitive to argument structures (Rohde et al., 2004). Nevertheless, it is worthwhile testing whether the meaning components that are brought out by syntactic alternations are also correlated to the neighboring words. In other words, Levin verbs may be distinguished on the dimension of neighboring words, in addition to argument structures. A test on this claim can help answer the question of whether verbs in the same Levin class also tend to share their neighboring words.

**Adapted co-occurrence (ACO)**: Conventional CO features generally adopt a stop list to filter out function words. However, some of the function words, prepositions in particular, are known to carry great amount of syntactic information that is related to lexical meanings of verbs (Schulte im Walde, 2003; Brew and Schulte im Walde, 2002; Joanis et al., 2007). In addition, whereas most verbs tend to put a strong selectional preference on their nominal arguments, they do not care much about the identity of the verbs in their verbal arguments. Based on these observations, we propose to adapt the conventional CO features by (1) keeping all prepositions (2) replacing all verbs in the neighboring contexts of each target verb with their part-of-speech tags. ACO features integrate at least some degree of syntactic information into the feature space.

**SCF+CO**: Another way to mix syntactic information with lexical information is to use subcategorization frames and co-occurrences together in hope that they are complementary to each other, and therefore yield better results for AVC.

## 4 Experiment Setup

### 4.1 Corpus

To collect each type of features, we use the Gigaword Corpus, which consists of samples of recent newswire text data collected from four distinct in-



ternational sources of English newswire.

## 4.2 Feature Extraction

We evaluate six different feature sets for their effectiveness in AVC: **SCF**, **DR**, **CO**, **ACO**, **SCF+CO**, and **JOANIS07**. **SCF** contains mainly syntactic information, whereas **CO** lexical information. The other four feature sets include both syntactic and lexical information.

**SCF** and **DR**: These more linguistically informed features are constructed based on the grammatical relations generated by the C&C CCG parser (Clark and Curran, 2007). Take *He broke the door with a hammer* as an example. The grammatical relations generated are given in table 2.

<i>he broke the door with a hammer.</i>
(det door.3 the.2)
( <b>dobj</b> _ broke.1 door.3)
(det hammer.6 a.5)
( <b>dobj</b> with.4 hammer.6)
( <b>iobj</b> broke.1 with.4)
( <b>ncsubj</b> broke.1 He.0 _)

Table 2: grammatical relations generated by the parser

We first build a lexicalized frame for the verb *break*: NP1(*he*)-V-NP2(*door*)-PP(*with:hammer*). This is done by matching each grammatical label onto one of the traditional syntactic constituents. The set of syntactic constituents we use is summarized in table 3.

constituent	remark
NP1	subject of the verb
NP2	object of the verb
NP3	indirect object of the verb
PPp	prepositional phrase
TO	infinitival clause
GER	gerund
THAT	sentential complement headed by <i>that</i>
WH	sentential complement headed by a <i>wh</i> -word
ADJP	adjective phrase
ADVP	adverb phrase

Table 3: Syntactic constituents used for building SCFs

Based on the lexicalized frame, we construct an SCF NP1-NP2-PP*with* for *break*. The set of DRs generated for *break* is [SUBJ(*he*), OBJ(*door*), PP(*with*), PP-*hammer*].

**CO**: These features are collected using a flat 4-word window, meaning that the 4 words to the

left/right of each target verb are considered potential CO features. However, we eliminate any CO features that are in a stopword list, which consists of about 200 closed class words including mainly prepositions, determiners, complementizers and punctuation. We also lemmatize each word using the English lemmatizer as described in Minnen et al. (2000), and use lemmas as features instead of words.

**ACO**: As mentioned before, we adapt the conventional CO features by (1) keeping all prepositions (2) replacing all verbs in the neighboring contexts of each target verb with their part-of-speech tags. (3) keeping words in the left window only if they are tagged as a nominal.

**SCF+CO**: We combine the SCF and CO features.

**JOANIS07**: We use the feature set proposed in Joanis et al. (2007), which consists of 224 features. We extract features on the basis of the output generated by the C&C CCG parser.

## 4.3 Verb Classes

Our experiments involve two separate sets of verb classes:

**Joanis15**: Joanis et al. (2007) manually selects pairs, or triples of classes to represent a range of distinctions that exist among the 15 classes they investigate. For example, some of the pairs/triples are syntactically dissimilar, while others show little syntactic distinction across the classes.

**Levin48**: Earlier work has focused only on a small set of verbs or a small number of verb classes. For example, Schulte im Walde (2000) uses 153 verbs in 30 classes, and Joanis et al. (2007) takes on 835 verbs and 15 verb classes. Since one of our primary goals is to identify a general feature space that is not specific to any class distinctions, it is of great importance to understand how the classification accuracy is affected when attempting to classify more verbs into a larger number of classes. In our automatic verb classification, we aim for a larger scale experiment. We select our experimental verb classes and verbs as follows: We start with all Levin 197 verb classes. We first remove all verbs that belong to at least two Levin classes. Next, we remove any verb that does not occur at least 100 times in the English Gigaword Corpus. All classes that are left with at least 10 verbs are chosen for our experi-

ment. This process yields 48 classes involving about 1,300 verbs. In our automatic verb classification experiment, we test the applicability of each feature set to distinctions among up to 48 classes<sup>1</sup>. To our knowledge, this is, by far, the largest investigation on English verb classification.

## 5 Machine Learning Method

### 5.1 Preprocessing Data

We represent the semantic space for verbs as a matrix of frequencies, where each row corresponds to a Levin verb and each column represents a given feature. We construct a semantic space with each feature set. Except for **JONAS07** which only contains 224 features, all the other feature sets lead to a very high-dimensional space. For instance, the semantic space with **CO** features contains over one million columns, which is too huge and cumbersome. One way to avoid these high-dimensional spaces is to assume that most of the features are irrelevant, an assumption adopted by many of the previous studies working with high-dimensional semantic spaces (Burgess and Lund, 1997; Pado and Lapata, 2007; Rohde et al., 2004). Burgess and Lund (1997) suggests that the semantic space can be reduced by keeping only the  $k$  columns (features) with the highest variance. However, Rohde et al. (2004) have found it is simpler and more effective to discard columns on the basis of feature frequency, with little degradation in performance, and often some improvement. Columns representing low-frequency features tend to be noisier because they only involve few examples. We therefore apply a simple frequency cutoff for feature selection. We only use features that occur with a frequency over some threshold in our data.

In order to reduce undue influence of outlier features, we employ the four normalization strategies in table 4, which help reduce the range of extreme values while having little effect on others (Rohde et al., 2004). The raw frequency ( $w_{v,f}$ ) of a verb  $v$  occurring with a feature  $f$  is replaced with the normal-

ized value ( $w'_{v,f}$ ), according to each normalization method. Our experiments show that using correlation for normalization generally renders the best results. The results reported below are obtained from using correlation for normalization.

	$w'_{v,f} =$
row	$\frac{w_{v,f}}{\sum_j w_{v,j}}$
column	$\frac{w_{v,f}}{\sum_i w_{i,f}}$
length	$\frac{w_{v,f}}{\sum_j w_{v,j}^2}^{1/2}$
correlation	$\frac{T w_{v,f} - \sum_j w_{v,j} \sum_i w_{i,f}}{(\sum_j w_{v,j} (T - \sum_j w_{v,j}) \sum_i w_{i,f} (T - \sum_i w_{i,f}))^{1/2}}$
	$T = \sum_i \sum_j w_{i,j}$

Table 4: Normalization techniques

To preprocess data, we first apply a frequency cutoff to our data set, and then normalize it using the correlation method. To find the optimal threshold for frequency cut, we consider each value between 0 and 10,000 at an interval of 500. In our experiments, results on training data show that performance declines more noticeably when the threshold is lower than 500 or higher than 10,000. For each task and feature set, we select the frequency cut that offers the best accuracy on the preprocessed training set according to  $k$ -fold stratified cross validation<sup>2</sup>.

### 5.2 Classifier

For all of our experiments, we use the software that implements the Bayesian multinomial logistic regression (a.k.a BMR). The software performs the so-called 1-of- $k$  classification (Madigan et al., 2005). BMR is similar to Maximum Entropy. It has been shown to be very efficient with handling large numbers of features and extremely sparsely populated matrices, which characterize the data we have for AVC<sup>3</sup>. To begin, let  $x = [x_1, \dots, x_j, \dots, x_d]^T$  be a vector of feature values characterizing a verb to be classified. We encode the fact that a verb belongs to a class  $k \in 1, \dots, K$  by a  $K$ -dimensional 0/1 valued vector  $y = (y_1, \dots, y_K)^T$ , where  $y_k = 1$  and all other coordinates are 0. Multinomial logistic regres-

<sup>1</sup>In our experiment, we only use monosemous verbs from these 48 verb classes. Due to the space limit, we do not list the 48 verb classes. The size of the most classes falls in the range between 10 to 30, with a couple of classes having a size over 100.

<sup>2</sup>10-fold for Joanis15 and 9-fold for Levin48. We use a balanced training set, which contains 20 verbs from each class in Joanis15, but only 9 verbs from each class in Levin48.

<sup>3</sup>We also tried Chang and Lin (2001)'s LIBSVM library for Support Vector Machines (SVMs), however, BMR generally outperforms SVMs.

sion is a conditional probability model of the form, parameterized by the matrix  $\beta = [\beta_1, \dots, \beta_K]$ . Each column of  $\beta$  is a parameter vector corresponding to one of the classes:  $\beta_k = [\beta_{k1}, \dots, \beta_{kd}]^T$ .

$$P(y_k = 1 | \beta_k, x) = \exp(\beta_k^T x) / \sum_{k_i} \exp(\beta_{k_i}^T x)$$

## 6 Results and Discussion

### 6.1 Evaluation Metrics

Following Joanis et al. (2007), we adopt a single evaluation measure - macro-averaged recall - for all of our classification tasks. As discussed below, since we always use balanced training sets for each individual task, it makes sense for our accuracy metric to give equal weight to each class. Macro-averaged recall treats each verb class equally, so that the size of a class does not affect macro-averaged recall. It usually gives a better sense of the quality of classification across all classes. To calculate macro-averaged recall, the recall value for each individual verb class has to be computed first.

$$recall = \frac{\text{no. of test verbs in class } c \text{ correctly labeled}}{\text{no. of test verbs in class } c}$$

With a recall value computed for each verb class, the macro-averaged recall can be defined by:

$$macro\text{-averaged recall} = \frac{1}{|C|} \sum_{c \in C} recall \text{ for class } c$$

$C$  : a set of verb classes

$c$  : an individual verb class

$|C|$  : the number of verb classes

### 6.2 Joanis15

With those manually-selected 15 classes, Joanis et al. (2007) conducts 11 classification tasks including six 2-way classifications, two 3-way classifications, one 6-way classification, one 8-way classification, and one 14-way classification. In our experiments, we replicate these 11 classification tasks using the proposed six different feature sets. For each classification task in this task set, we randomly select 20 verbs from each class as the training set. We

repeat this process 10 times for each task. The results reported for each task is obtained by averaging the results of the 10 trials. Note that for each trial, each feature set is trained and tested on the same training/test split.

The results for the 11 classification tasks are summarized in table 5. We provide a chance baseline and the accuracy reported in Joanis et al. (2007)<sup>4</sup> for comparison of our results. A few points are worth noting:

- Although widely used for AVC, **SCF**, at least when used alone, is not the most effective feature set. Our experiments show that the performance achieved by using **SCF** is generally worse than using the feature sets that mix syntactic and lexical information. As a matter of fact, it even loses to the simplest feature set **CO** on 4 tasks, including the 14-way task.
- The two feature sets (**DR**, **SCF+CO**) we propose that combine syntactic and lexical information generally perform better than those feature sets (**SCF**, **CO**) that only include syntactic or lexical information. Although there is not a clear winner, **DR** and **SCF+CO** generally outperform other feature sets, indicating that they are effective ways for combining syntactic and lexical information. In particular, these two feature sets perform comparatively well on the tasks that involve more classes (e.g. 14-way), exhibiting the tendency to scale well with larger number of verb classes and verbs. Another feature set that combines syntactic and lexical information, **ACO**, which keeps function words in the feature space to preserve syntactic information, outperforms the conventional **CO** on the majority of tasks. All these observations suggest that how to mix syntactic and lexical information is one of keys to an improved verb classification.
- Although **JOANIS07** also combines syntactic and lexical information, its performance is not comparable to that of other feature sets that mix syntactic and lexical information. In fact, **SCF**

<sup>4</sup>Joanis et al. (2007) is different from our experiments in that they use a chunker for feature extraction and the Support Vector Machine for classification.

Experimental Task	Random Baseline	As Reported in Joanis et al. (2007)	Feature Set					
			SCF	DR	CO	ACO	SCF+CO	JOANIS07
1) Benefactive/Recipient	50	86.4	88.6	88.4	88.2	89.1	<b>90.7</b>	88.9
2) Admire/Amuse	50	93.9	96.7	<b>97.5</b>	92.1	90.5	96.4	96.6
3) Run/Sound	50	86.8	85.4	89.6	<b>91.8</b>	90.2	90.5	87.1
4) Light/Sound	50	75.0	74.8	<b>90.8</b>	86.9	89.7	88.8	82.1
5) Cheat/Steal	50	76.5	77.6	<b>80.6</b>	72.1	75.5	77.8	76.4
6) Wipe/Steal	50	80.4	<b>84.8</b>	80.6	79.0	79.4	84.4	83.9
7) Spray/Fill/Putting	33.3	65.6	73.0	72.8	59.6	66.6	<b>73.8</b>	69.6
8) Run/State Change/Object drop	33.3	74.2	74.8	77.2	76.9	77.6	<b>80.5</b>	75.5
9) Cheat/Steal/Wipe/Spray/Fill/Putting	16.7	64.3	64.9	<b>65.1</b>	54.8	59.1	65.0	64.3
10) 9)/Run/Sound	12.5	61.7	62.3	65.8	55.7	60.8	<b>66.9</b>	63.1
11) 14-way (all except Benefactive)	7.1	58.4	56.4	65.7	57.5	59.6	<b>66.3</b>	57.2

Table 5: Experimental results for Joanis15 (%)

and **JOANIS07** yield similar accuracy in our experiments, which agrees with the findings in Joanis et al. (2007) (compare table 1 and 5).

### 6.3 Levin48

Recall that one of our primary goals is to identify the feature set that is generally applicable and scales well while we attempt to classify more verbs into a larger number of classes. If we could exhaust all the possible  $n$ -way ( $2 \leq n \leq 48$ ) classification tasks with the 48 Levin classes we will investigate, it will allow us to draw a firmer conclusion about the general applicability and scalability of a particular feature set. However, the number of classification tasks grows really huge when  $n$  takes on certain value (e.g.  $n = 20$ ). For our experiments, we set  $n$  to be 2, 5, 10, 20, 30, 40, or 48. For the 2-way classification, we perform all the possible 1,028 tasks. For the 48-way classification, there is only one possible task. We randomly select 100  $n$ -way tasks each for  $n = 5, 10, 20, 30, 40$ . We believe that this series of tasks will give us a reasonably good idea of whether a particular feature set is generally applicable and scales well.

The smallest classes in Levin48 have only 10 verbs. We therefore reduce the number of training verbs to 9 for each class. For each  $n = 2, 5, 10, 20, 30, 40, 48$ , we will perform certain number of  $n$ -way classification tasks. For each  $n$ -way task, we randomly select 9 verbs from each class as training data, and repeat this process 10 times. The accuracy for each  $n$ -way task is then computed by averaging the results from these 10 trials. The accuracy reported for the overall  $n$ -way classification for each selected  $n$ , is obtained by averaging the results from each in-

dividual  $n$ -way task for that particular  $n$ . Again, for each trial, each feature set is trained and tested on the same training/test split.

The results for Levin48 are presented in table 6, which clearly reveals the general applicability and scalability of each feature set.

- Results from Levin48 reconfirm our finding that **SCF** is not the most effective feature set for AVC. Although it achieves the highest accuracy on the 2-way classification, its accuracy drops drastically as  $n$  gets bigger, indicating that SCF does not scale as well as other feature sets when dealing with larger number of verb classes. On the other hand, the co-occurrence feature (**CO**), which is believed to convey only lexical information, outperforms **SCF** on every  $n$ -way classification when  $n \geq 10$ , suggesting that verbs in the same Levin classes tend to share their neighboring words.
- The three feature sets we propose that combine syntactic and lexical information generally scale well. Again, **DR** and **SCF+CO** generally outperform all other feature sets on all  $n$ -way classifications, except the 2-way classification. In addition, **ACO** achieves a better performance on every  $n$ -way classification than **CO**. Although **SCF** and **CO** are not very effective when used individually, they tend to yield the best performance when combined together.
- Again, **JOANIS07** does not match the performance of other feature sets that combine both syntactic and lexical information, but yields similar accuracy as **SCF**.

Experimental Task	No of Tasks	Random Baseline	Feature Set					
			SCF	DR	CO	ACO	SCF+CO	JOANIS07
2-way	1,028	50	<b>84.0</b>	83.4	77.8	80.9	82.9	82.4
5-way	100	20	71.9	76.4	70.4	73.0	<b>77.3</b>	72.2
10-way	100	10	65.8	<b>73.7</b>	68.8	71.2	72.8	65.9
20-way	100	5	51.4	65.1	58.8	60.1	<b>65.8</b>	50.7
30-way	100	3.3	46.7	56.9	48.6	51.8	<b>57.8</b>	47.1
40-way	100	2.5	43.6	54.8	47.3	49.9	<b>55.1</b>	44.2
48-way	1	2.2	39.1	51.6	42.4	46.8	<b>52.8</b>	38.9

Table 6: Experimental results for Levin48 (%)

## 6.4 Further Discussion

Previous studies on AVC have focused on using SCFs. Our experiments reveal that SCFs, at least when used alone, compare poorly to the feature sets that mix syntactic and lexical information. One explanation for the poor performance could be that we use all the frames generated by the CCG parser in our experiment. A better way of doing this would be to use some expert-selected SCF set. Levin classifies English verbs on the basis of 78 SCFs, which should, at least in principle, be good at separating verb classes. To see if Levin-selected SCFs are more effective for AVC, we match each SCF generated by the C&C CCG parser (**CCG-SCF**) to one of 78 Levin-defined SCFs, and refer to the resulting SCF set as **unfiltered-Levin-SCF**. Following studies on automatic SCF extraction (Brent, 1993), we apply a statistical test (Binomial Hypothesis Test) to the **unfiltered-Levin-SCF** to filter out noisy SCFs, and denote the resulting SCF set as **filtered-Levin-SCF**. We then perform the 48-way task (one of Levin48) with these two different SCF sets. Recall that using **CCG-SCF** gives us a macro-averaged recall of 39.1% on the 48-way task. Our experiments show that using **unfiltered-Levin-SCF** and **filtered-Levin-SCF** raises the accuracy to 39.7% and 40.3% respectively. Although a little performance gain has been obtained by using expert-defined SCFs, the accuracy level is still far below that achieved by using a feature set that combines syntactic and semantic information. In fact, even the simple co-occurrence feature (CO) yields a better performance (42.4%) than these Levin-selected SCF sets.

## 7 Conclusion and Future Work

We have performed a wide range of experiments to identify which features are most informative in

AVC. Our conclusion is that both syntactic and lexical information are useful for verb classification. Although neither **SCF** nor **CO** performs well on its own, a combination of them proves to be the most informative feature for this task. Other ways of mixing syntactic and lexical information, such as **DR**, and **ACO**, work relatively well too. What makes these mixed feature sets even more appealing is that they tend to scale well in comparison to **SCF** and **CO**. In addition, these feature sets are devised on a general level without relying on any knowledge about specific classes, thus potentially applicable to a wider range of class distinctions. Assuming that Levin’s analysis is generally applicable across languages in terms of the linking of semantic arguments to their syntactic expressions, these mixed feature sets are potentially useful for building verb classifications for other languages.

For our future work, we aim to test whether an automatically created verb classification can be beneficial to other NLP tasks. One potential application of our verb classification is parsing. Lexicalized PCFGs (where head words annotate phrasal nodes) have proved a key tool for high performance PCFG parsing, however its performance is hampered by the sparse lexical dependency exhibited in the Penn Treebank. Our experiments on verb classification have offered a class-based approach to alleviate data sparsity problem in parsing. It is our goal to test whether this class-based approach will lead to an improved parsing performance.

## 8 Acknowledgments

This study was supported by NSF grant 0347799. We are grateful to Eric Fosler-Lussier, Detmar Meurers, Mike White and Kirk Baker for their valuable comments.

## References

- Brent, M. (1993). From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, 19(3):243–262.
- Brew, C. and Schulte im Walde, S. (2002). Spectral clustering for German verbs. In *Proceedings of the 2002 Conference on EMNLP*, pages 117–124.
- Burgess, C. and Lund, K. (1997). Modelling parsing constraints with high-dimensional context space. *Language and Cognitive Processes*, 12(3):177–210.
- Chang, C. and Lin, C. (2001). LIBSVM: A library for support vector machines. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Clark, S. and Curran, J. (2007). Formalism-independent parser evaluation with CCG and Depbank. In *Proceedings of the 45th Annual Meeting of ACL*, pages 248–255.
- Dowty, D. (1991). Thematic proto-roles and argument selection. *Language*, 67:547–619.
- Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic role. *Computational Linguistics*, 28(3):245–288.
- Goldberg, A. (1995). *Constructions*. University of Chicago Press, Chicago, 1st edition.
- Green, G. (1974). *Semantics and Syntactic Regularity*. Indiana University Press, Bloomington.
- Habash, N., Dorr, B., and Traum, D. (2003). Hybrid natural language generation from lexical conceptual structures. *Machine Translation*, 18(2):81–128.
- Joanis, E. (2002). Automatic verb classification using a general feature space. Master’s thesis, University of Toronto.
- Joanis, E., Stevenson, S., and James, D. (2007). A general feature space for automatic verb classification. *Natural Language Engineering*, 1:1–31.
- Korhonen, A. (2002). *Subcategorization Acquisition*. PhD thesis, Cambridge University.
- Korhonen, A. and Briscoe, T. (2004). Extended lexical-semantic classification of english verbs. In *Proceedings of the 2004 HLT/NAACL Workshop on Computational Lexical Semantics*, pages 38–45, Boston, MA.
- Korhonen, A., Krymolowski, Y., and Collier, N. (2006). Automatic classification of verbs in biomedical texts. In *Proceedings of the 21st International Conference on COLING and 44th Annual Meeting of ACL*, pages 345–352, Sydney, Australia.
- Korhonen, A., Krymolowski, Y., and Marx, Z. (2003). Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of the 41st Annual Meeting of ACL*, pages 48–55, Sapporo, Japan.
- Lapata, M. and Brew, C. (2004). Verb class disambiguation using informative priors. *Computational Linguistics*, 30(1):45–73.
- Levin, B. (1993). *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, 1st edition.
- Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on COLING and 36th Annual Meeting of ACL*.
- Madigan, D., Genkin, A., Lewis, D., and Fradkin, D. (2005). Bayesian Multinomial Logistic Regression for Author Identification. *DIMACS Technical Report*.
- McCarthy, D., Koeling, R., Weeds, J., and Carroll, J. (2004). Finding predominant senses in untagged text. In *Proceedings of the 42nd Annual Meeting of ACL*, pages 280–287.
- Merlo, P. and Stevenson, S. (2001). Automatic verb classification based on statistical distribution of argument structure. *Computational Linguistics*, 27(3):373–408.
- Minnen, G., Carroll, J., and Pearce, D. (2000). Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.
- Pado, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.
- Rohde, D., Gonnerman, L., and Plaut, D. (2004). An improved method for deriving word meaning from lexical co-occurrence. <http://dlt4.mit.edu/dr/COALS>.
- Schulte im Walde, S. (2000). Clustering verbs semantically according to alternation behavior. In *Proceedings of the 18th International Conference on COLING*, pages 747–753.
- Schulte im Walde, S. (2003). Experiments on the choice of features for learning verb classes. In *Proceedings of the 10th Conference of EACL*, pages 315–322.
- Swier, R. and Stevenson, S. (2004). Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on EMNLP*, pages 95–102.
- Tsang, V., Stevenson, S., and Merlo, P. (2002). Crosslinguistic transfer in automatic verb classification. In *Proceedings of the 19th International Conference on COLING*, pages 1023–1029, Taiwan, China.

# Collecting a Why-question corpus for development and evaluation of an automatic QA-system

Joanna Mrozinski

Edward Whittaker

Sadaoki Furui

Department of Computer Science

Tokyo Institute of Technology

2-12-1-W8-77 Ookayama, Meguro-ku

Tokyo 152-8552 Japan

{mrozinsk,edw,furui}@furui.cs.titech.ac.jp

## Abstract

Question answering research has only recently started to spread from short factoid questions to more complex ones. One significant challenge is the evaluation: manual evaluation is a difficult, time-consuming process and not applicable within efficient development of systems. Automatic evaluation requires a corpus of questions and answers, a definition of what is a correct answer, and a way to compare the correct answers to automatic answers produced by a system. For this purpose we present a Wikipedia-based corpus of Why-questions and corresponding answers and articles. The corpus was built by a novel method: paid participants were contacted through a Web-interface, a procedure which allowed dynamic, fast and inexpensive development of data collection methods. Each question in the corpus has several corresponding, partly overlapping answers, which is an asset when estimating the correctness of answers. In addition, the corpus contains information related to the corpus collection process. We believe this additional information can be used to post-process the data, and to develop an automatic approval system for further data collection projects conducted in a similar manner.

## 1 Introduction

Automatic question answering (QA) is an alternative to traditional word-based search engines. Instead of returning a long list of documents more or less related to the query parameters, the aim of a QA system is to isolate the exact answer as accurately as

possible, and to provide the user only a short text clip containing the required information.

One of the major development challenges is evaluation. The conferences such as TREC<sup>1</sup>, CLEF<sup>2</sup> and NTCIR<sup>3</sup> have provided valuable QA evaluation methods, and in addition produced and distributed corpora of questions, answers and corresponding documents. However, these conferences have focused mainly on fact-based questions with short answers, so called factoid questions. Recently more complex tasks such as list, definition and discourse-based questions have also been included in TREC in a limited fashion (Dang et al., 2007). More complex how- and why-questions (for Asian languages) were also included in the NTCIR07, but the provided data comprised only 100 questions, of which some were also factoids (Fukumoto et al., 2007). Not only is the available non-factoid data quite limited in size, it is also questionable whether the data sets are usable in development outside the conferences. Lin and Katz (2006) suggest that training data has to be more precise, and, that it should be collected, or at least cleaned, manually.

Some corpora of why-questions have been collected manually: corpora described in (Verberne et al., 2006) and (Verberne et al., 2007) both comprise fewer than 400 questions and corresponding answers (one or two per question) formulated by native speakers. However, we believe one answer per question is not enough. Even with factoid questions it is sometimes difficult to define what is a correct

<sup>1</sup><http://trec.nist.gov/>

<sup>2</sup><http://www.clef-campaign.org/>

<sup>3</sup><http://research.nii.ac.jp/ntcir/>

answer, and complex questions result in a whole new level of ambiguity. Correctness depends greatly on the background knowledge and expectations of the person asking the question. For example, a correct answer to the question “Why did Mr. X take Ms. Y to a coffee shop?” could be very different depending on whether we knew that Mr. X does not drink coffee or that he normally drinks it alone, or that Mr. X and Ms. Y are known enemies.

The problem of several possible answers and, in consequence, automatic evaluation has been tackled for years within another field of study: automatic summarisation (Hori et al., 2003; Lin and Hovy, 2003). We believe that the best method of providing “correct” answers is to do what has been done in that field: combine a multitude of answers to ensure both diversity and consensus among the answers.

Correctness of an answer is also closely related to the required level of detail. The Internet FAQ pages were successfully used to develop QA-systems (Jijikoun and de Rijke, 2005; Soricut and Brill, 2006), as have the human-powered question sites such as Answers.com, Yahoo Answers and Google Answers, where individuals can post questions and receive answers from peers (Mizuno et al., 2007). Both resources can be assumed to contain adequately error-free information. FAQ pages are created so as to answer typical questions well enough that the questions do not need to be repeated. Question sites typically rank the answers and offer bonuses for people providing good ones. However, both sites suffer from excess of information. FAQ-pages tend to also answer questions which are not asked, and also contain practical examples. Human-powered answers often contain unrelated information and discourse-like elements. Additionally, the answers do not always have a connection to the source material from which they could be extracted.

One purpose of our project was to take part in the development of QA systems by providing the community with a new type of corpus. The corpus includes not only the questions with multiple answers and corresponding articles, but also certain additional information that we believe is essential to enhance the usability of the data.

In addition to providing a new QA corpus, we hope our description of the data collection process will provide insight, resources and motivation for

further research and projects using similar collection methods. We collected our corpus through Amazon Mechanical Turk service <sup>4</sup> (*MTurk*). The MTurk infrastructure allowed us to distribute our tasks to a multitude of workers around the world, without the burden of advertising. The system also allowed us to test the workers suitability, and to reward the work without the bureaucracy of employment. To our knowledge, this is the first time that the MTurk service has been used in equivalent purpose.

We conducted the data collection in three steps: generation, answering and rephrasing of questions. The workers were provided with a set of Wikipedia articles, based on which the questions were created and the answers determined by sentence selection. The WhyQA-corpus consists of three parts: original questions along with their rephrased versions, 8-10 partly overlapping answers for each question, and the Wikipedia articles including the ones corresponding to the questions. The WhyQA-corpus is in XML-format and can be downloaded and used under the GNU Free Documentation License from [www.furui.cs.titech.ac.jp/](http://www.furui.cs.titech.ac.jp/).

## 2 Setup

Question-answer pairs have previously been generated for example by asking workers to both ask a question and then answer it based on a given text (Verberne et al., 2006; Verberne et al., 2007). We decided on a different approach for two reasons. Firstly, based on our experience such an approach is not optimal in the MTurk framework. The tasks that were welcomed by workers required a short attention span, and reading long texts was negatively received with many complaints, sloppy work and slow response times. Secondly, we believe that the aforementioned approach can produce unnatural questions that are not actually based on the information need of the workers.

We divided the QA-generation task into two phases: question-generation (*QGenHIT*) and answering (*QAHIT*). We also trimmed the amount of the text that the workers were required to read to create the questions. These measures were taken both in order to lessen the cognitive burden of the task

<sup>4</sup><http://www.mturk.com>



and to produce more natural questions.

In the first phase the workers generated the questions based on a part of Wikipedia article. The resulting questions were then uploaded to the system as new HITs with the corresponding articles, and answered by available (different) workers. Our hypothesis is that the questions are more natural if their answer is not known at the time of the creation.

Finally, in an additional third phase, 5 rephrased versions of each question were created in order to gain variation (*QRepHIT*). The data quality was ensured by requiring the workers to achieve a certain result from a test (or a *Qualification*) before they could work on the aforementioned tasks.

Below we explain the MTurk system, and then our collection process in detail.

## 2.1 Mechanical Turk

Mechanical Turk is a Web-based service, offered by Amazon.com, Inc. It provides an API through which employers can obtain a connection to people to perform a variety of simple tasks. With tools provided by Amazon.com, the employer creates tasks, and uploads them to the MTurk Web-site. Workers can then browse the tasks and, if they find them profitable and/or interesting enough, work on them. When the tasks are completed, the employer can download the results, and accept or reject them. Some key concepts of the system are listed below, with short descriptions of the functionality.

- **HIT** Human Intelligence Task, the unit of a payable chore in MTurk.
- **Requester** An “employer”, creates and uploads new *HITs* and rewards the *workers*. Requesters can upload simple *HITs* through the MTurk Requester web site, and more complicated ones through the MTurk Web Service APIs.
- **Worker** An “employee”, works on the hits through the MTurk Workers’ web site.
- **Assignment.** One *HIT* consists of one or more assignments. One worker can complete a single *HIT* only once, so if the requester needs multiple results per *HIT*, he needs to set the assignment-count to the desired figure. A *HIT* is considered completed when all the assignments have been completed.

- **Rewards** At upload time, each *HIT* has to be assigned a fixed reward, that cannot be changed later. Minimum reward is \$0.01. Amazon.com collects a 10% (or a minimum of \$0.05) service fee per each paid reward.

- **Qualifications** To improve the data quality, a *HIT* can also be attached to certain tests, “qualifications” that are either system-provided or created by the requester. An example of a system-provided qualification is the average approval ratio of the worker.

Even if it is possible to create tests that workers have to pass before being allowed to work on a *HIT* so as to ensure the worker’s ability, it is impossible to test the motivation (for instance, they cannot be interviewed). Also, as they are working through the Web, their working conditions cannot be controlled.

## 2.2 Collection process

The document collection used in our research was derived from the Wikipedia XML Corpus by Denoyer and Gallinari (2006). We selected a total of 84 articles, based on their length and contents. A certain length was required so that we could expect the article to contain enough interesting material to produce a wide selection of natural questions. The articles varied in topic, degree of formality and the amount of details; from “Horror film” and “Christmas worldwide” to “G-Man (Half-Life)” and “History of London”. Articles consisting of bulleted lists were removed, but filtering based on the topic of the article was not performed. Essentially, the articles were selected randomly.

### 2.2.1 QGenHIT

The first phase of the question-answer generation was to generate the questions. In QGenHIT we presented the worker with only part of a Wikipedia article, and instructed them to think of a why-question that they felt could be answered based on the original, whole article which they were not shown. This approach was expected to lead to natural curiosity and questions. Offering too little information would have lead to many questions that would finally be left unanswered, and it also did not give the workers enough to work on. Giving too much information

<b>Qualification</b>	The workers were required to pass a test before working on the HITs.
<b>QGenHIT</b>	Questions were generated based on partial Wikipedia articles. These questions were then used to create the QAHITs.
<b>QAHIT</b>	Workers were presented with a question and a corresponding article. The task was to answer the questions (if possible) through sentence selection.
<b>QRepHIT</b>	To ensure variation in the questions, each question was rephrased by 5 different workers.

Table 1: Main components of the corpus collection process.

<b>Article topic: Fermi paradox</b>
<b>Original question</b> Why is the moon crucial to the rare earth hypothesis?
<b>Rephrased Q 1</b> How does the rare earth theory depend upon the moon?
<b>Rephrased Q 2</b> What makes the moon so important to rare earth theory?
<b>Rephrased Q 3</b> What is the crucial regard for the moon in the rare earth hypothesis?
<b>Rephrased Q 4</b> Why is the moon so important in the rare earth hypothesis?
<b>Rephrased Q 5</b> What makes the moon necessary, in regards to the rare earth hypothesis?
<b>Answer 1. Sentence ids: 20,21. Duplicates: 4.</b> The moon is important because its gravitational pull creates tides that stabilize Earth’s axis. Without this stability, its variation, known as precession of the equinoxes, could cause weather to vary so dramatically that it could potentially suppress the more complex forms of life.
<b>Answer 2. Sentence ids: 18,19,20. Duplicates: 2.</b> The popular Giant impact theory asserts that it was formed by a rare collision between the young Earth and a Mars-sized body, usually referred to as Orpheus or Theia, approximately 4.45 billion years ago. The collision had to occur at a precise angle, as a direct hit would have destroyed the Earth, and a shallow hit would have deflected the Mars-sized body. The moon is important because its gravitational pull creates tides that stabilize Earth’s axis.
<b>Answer 3. Sentence ids: 20,21,22. Duplicates: 2.</b> The moon is important because its gravitational pull creates tides that stabilize Earth’s axis. Without this stability, its variation, known as precession of the equinoxes, could cause weather to vary so dramatically that it could potentially suppress the more complex forms of life. The heat generated by the Earth/Theia impact, as well as subsequent Lunar tides, may have also significantly contributed to the total heat budget of the Earth’s interior, thereby both strengthening and prolonging the life of the dynamos that generate Earth’s magnetic field Dynamo 1.
<b>Answer 4. Sentence ids: 18,20,21. No duplicates.</b> The popular Giant impact theory asserts that it was formed by a rare collision between the young Earth and a Mars-sized body, usually referred to as Orpheus or Theia, approximately 4.45 billion years ago. The moon is important because its gravitational pull creates tides that stabilize Earth’s axis. Without this stability, its variation, known as precession of the equinoxes, could cause weather to vary so dramatically that it could potentially suppress the more complex forms of life.
<b>Answer 5. Sentence ids: 18,21. No duplicates.</b> The popular Giant impact theory asserts that it was formed by a rare collision between the young Earth and a Mars-sized body, usually referred to as Orpheus or Theia, approximately 4.45 billion years ago. Without this stability, its variation, known as precession of the equinoxes, could cause weather to vary so dramatically that it could potentially suppress the more complex forms of life.

Table 2: Data example: Question with rephrased versions and answers.

(long excerpts from the articles) was severely disliked among the workers simply because it took a long time to read.

We finally settled on a solution where the partial content consisted of the title and headers of the article, along with the first sentences of each paragraph. The instructions to the questions demanded rigidly that the question starts with the word “Why”, as it was surprisingly difficult to explain what we meant by why-questions if the question word was not fixed.

The reward per HIT was \$0.04, and 10 questions were collected for each article. We did not force the questions to be different, and thus in the later phase some of the questions were removed manually as they were deemed to mean exactly the same thing. However, there were less than 30 of these duplicate questions in the whole data set.

### 2.2.2 QAHIT

After generating the questions based on partial articles, the resulting questions were uploaded to the system as HITs. Each of these QAHITs presented a single question with the corresponding original article. The worker’s task was to select either 1-3 sentences from the text, or a No-answer-option (*NoA*). Sentence selection was conducted with Javascript functionality, so the workers had no chance to include freely typed information within the answer (although a comment field was provided). The reward per HIT was \$0.06. At the beginning, we collected 10 answers per question, but we cut that down to 8 because the HITs were not completed fast enough.

The workers for QAHITs were drawn from the same pool as the workers for QGenHIT, and it was possible for the workers to answer the questions they had generated themselves.

### 2.2.3 QRepHIT

As the final step 5 rephrased versions of each question were generated. This was done to compensate the rigid instructions of the QGenHIT and to ensure variation in the questions. We have not yet measured how well the rephrased questions match the answers of their original versions. In the final QRepHIT questions were grouped into groups of 5. Each HIT consisted of 5 assignments, and a \$0.05 reward was offered for each HIT.

QRepHIT required the least amount of design and

trials, and workers were delighted with the task. The HITs were completed fast and well even in the case when we accidentally uploaded a set of HITs with no reward.

As with QAHIT, the worker pool for creating and rephrasing questions was the same. The questions were rephrased by their creator in 4 cases.

## 2.3 Qualifications

To improve the data quality, we used the qualifications to test the workers. For the QGenHITs we only used the system-provided “HIT approval rate”-qualification. Only workers whose previous work had been approved in 80% of the cases were able to work on our HITs.

In addition to the system-provided qualification, we created a why-question-specific qualification. The workers were presented with 3 questions, and they were to answer each by either selecting 1-3 most relevant sentences from a list of about 10 sentences, or by deciding that there is no answer present. The possible answer-sentences were divided into groups of essential, OK and wrong, and one of the questions did quite clearly have no answer. The scoring was such that it was impossible to get approved results if not enough essential sentences were included. Selecting sentences from the OK-group only was not sufficient, and selecting sentences from the wrong-group was penalized. A minimum score per question was required, but also the total score was relevant – component scores could compensate each other up to a point. However, if the question with no answer was answered, the score could not be of an approvable level. This qualification was, in addition to the minimum HIT approval rate of 80%, a prerequisite for both the QRepHITs and the QAHITs.

A total of 2355 workers took the test, and 1571 (67%) of them passed it, thus becoming our available worker pool. However, in the end the actual number of different workers was only 173.

Examples of each HIT, their instructions and the Qualification form are included in the final corpus. The collection process is summarised in Table 1.

### 3 Corpus description

The final corpus consists of questions with their rephrased versions and answers. There are total of 695 questions, of which 159 were considered unanswerable based on the articles, and 536 that have 8-10 answers each. The total cost of producing the corpus was about \$350, consisting of \$310 paid in workers rewards and \$40 in Mechanical Turk fees, including all the trials conducted during the development of the final system.

Also included is a set of Wikipedia documents (*WikiXML*, about 660 000 articles or 670MB in compressed format), including the ones corresponding to the questions (84 documents). The source of WikiXML is the English part of the Wikipedia XML Corpus by Denoyer and Gallinari (2006). In the original data some of the HTML-structures like lists and tables occurred within sentences. Our sentence-selection approach to QA required a more fine-grained segmentation and for our purpose, much of the HTML-information was redundant anyway. Consequently we removed most of the HTML-structures, and the table-cells, list-items and other similar elements were converted into sentences. Apart from sentence-information, only the section-title information was maintained. Example data is shown in Table 2.

#### 3.1 Task-related information

Despite the Qualifications and other measures taken in the collection phase of the corpus, we believe the quality of the data remains open to question. However, the Mechanical Turk framework provided additional information for each assignment, for example the time workers spent on the task. We believe this information can be used to analyse and use our data better, and have included it in the corpus to be used in further experiments.

- **Worker Id** Within the MTurk framework, each worker is assigned a unique id. Worker id can be used to assign a reliability-value to the workers, based on the quality of their previous work. It was also used to examine whether the same workers worked on the same data in different phases: Of the original questions, only 7 were answered and 4 other rephrased by the same worker they were created by. However, it has

to be acknowledged that it is also possible for one worker to have had several accounts in the system, and thus be working under several different worker ids.

- **Time On Task** The MTurk framework also provides the requester the time it took for the worker to complete the assignment after accepting it. This information is also included in the corpus, although it is impossible to know precisely how much time the workers actually spent on each task. For instance, it is possible that one worker had several assignments open at the same time, or that they were not concentrating fully on working on the task. A high value of Time On Task thus does not necessarily mean that the worker actually spent a long time on it. However, a low value indicates that he/she did only spend a short time on it.
- **Reward** Over the period spent collecting the data, we changed the reward a couple of times to speed up the process. The reward is reported per HIT.
- **Approval Status** Within the collection process we encountered some clearly unacceptable work, and rejected it. The rejected work is also included in the corpus, but marked as rejected. The screening process was by no means perfect, and it is probable that some of the approved work should have been rejected.
- **HIT id, Assignment id, Upload Time** HIT and assignment ids and original upload times of the HITs are provided to make it possible to retrace the collection steps if needed.
- **Completion Time** Completion time is the timestamp of the moment when the task was completed by a worker and returned to the system. The time between the completion time and the upload time is presumably highly dependent on the reward, and on the appeal of the task in question.

#### 3.2 Quality experiments

As an example of the post-processing of the data, we conducted some preliminary experiments on the answer agreement between workers.

Out of the 695 questions, 159 were filtered out in the first part of QAHIT. We first uploaded only 3 assignments, and the questions that 2 out of 3 workers deemed unanswerable were filtered out. This left 536 questions which were considered answered, each one having 8-10 answers from different workers. Even though in the majority of cases (83% of the questions) one of the workers replied with the NoA, the ones that answered did agree up to a point: of all the answers, 72% were such that all of their sentences were selected by at least two different workers. On top of this, an additional 17% of answers shared at least one sentence that was selected by more than one worker.

To understand the agreement better, we also calculated the average agreement of selected sentences based on sentence ids and N-gram overlaps between the answers. In both of these experiments, only those 536 questions that were considered answerable were included.

### 3.2.1 Answer agreement on sentence ids

As the questions were answered by means of sentence selection, the simplest method to check the agreement between the workers was to compare the ids of the selected sentences. The agreement was calculated as follows: each answer was compared to all the other answers for the same question. For each case, the agreement was defined as  $Agreement = \frac{CommonIds}{AllIds}$ , where *CommonIds* is the number of sentence ids that existed in both answers, and *AllIds* is the number of different ids in both answers. We calculated the overall average agreement ratio (*Total Avg*) and the average of the best matches between two assignments within one HIT (*Best Match*). We ran the test for two data sets: The most typical case of the workers cheating was to mark the question unanswerable. Because of this the first data set included only the real answers, and the NoAs were removed (*NoA not included*, 3872 answers). If an answer was compared with a NoA, the agreement was 0, and if two NoAs were compared, the agreement was 1. We did, however, also include the figures for the whole data set (*NoA included*, 4638 answers). The results are shown in Table 3.

The Best Match -results were quite high compared to the Total Avg. From this we can conclude

	Total Avg	Best Match
NoA not included	0.39	0.68
NoA included	0.34	0.68

Table 3: Answer agreement based on sentence ids.

that in the majority of cases, there was at least one quite similar answer among those for that HIT. However, comparing the sentence ids is only an indicative measure, and it does not tell the whole story about agreement. For each document there may exist several separate sentences that contain the same kind of information, and so two answers can be alike even though the sentence ids do not match.

### 3.2.2 Answer agreement based on ROUGE

Defining the agreement over several passages of texts has for a long time been a research problem within the field of automatic summarisation. For each document it is possible to create several summarisations that can each be considered correct. The problem has been approached by using the ROUGE-metric: calculating the N-gram overlap between manual, “correct” summaries, and the automatic summaries. ROUGE has been proven to correlate well with human evaluation (Lin and Hovy, 2003).

Overlaps of higher order N-grams are more usable within speech summarisation as they take the grammatical structure and fluency of the summary into account. When selecting sentences, this is not an issue, so we decided to use only unigram and bigram counts (Table 4: *R-1*, *R2*), as well as the skip-bigram values (*R-SU*) and the longest common N-gram metric R-L. We calculated the figures for two data sets in the same way as in the case of sentence id agreement. Finally, we set a lower bound for the results by comparing the answers to each other randomly (the NoAs were also included).

The final F-measures of the ROUGE results are presented in Table 4. The figures vary from 0.37 to 0.56 for the first data set, and from 0.28 to 0.42 to the second. It is debatable how the results should be interpreted, as we have not defined a theoretical upper bound to the values, but the difference to the randomised results is substantial. In the field of automatic summarisation, the overlap of the automatic

results and corresponding manual summarisations is generally much lower than the overlap between our answers (Chali and Kolla, 2004). However, it is difficult to draw detailed conclusions based on comparison between these two very different tasks.

	R-1	R-2	R-SU	R-L
NoA not included	0.56	0.46	0.37	0.52
NoA included	0.42	0.35	0.28	0.39
Random Answers	0.13	0.01	0.02	0.09

Table 4: Answer agreement: ROUGE-1, -2, -SU and -L.

The sentence agreement and ROUGE-figures do not tell us much by themselves. However, they are an example of a procedure that can be used to post-process the data and in further projects of similar nature. For example, the ROUGE similarity could be used in the data collection phase as a tool of automatic approval and rejection of workers’ assignments.

#### 4 Discussion and future work

During the initial trials of data collection we encountered some unexpected phenomena. For example, increasing the reward did have a positive effect in reducing the time it took for HITs to be completed, however it did not correlate in desirable way with data quality. Indeed the quality actually decreased with increasing reward. We believe that this unexpected result is due to the distributed nature of the worker pool in Mechanical Turk. Clearly the motivation of some workers is other than monetary reward. Especially if the HIT is interesting and can be completed in a short period of time, it seems that there are people willing to work on them even for free.

MTurk requesters cannot however rely on this voluntary workforce. From MTurk Forums it is clear that some of the workers rely on the money they get from completing the HITs. There seems to be a critical reward-threshold after which the “real workforce”, i.e. workers who are mainly interested in performing the HITs as fast as possible, starts to participate. When the motivation changes from voluntary participation to maximising the monetary gain, the quality of the obtained results often understandably suffers.

It would be ideal if a requester could rely on the voluntary workforce alone for results, but in many cases this may result either in too few workers and/or too slow a rate of data acquisition. Therefore it is often necessary to raise the reward and rely on efficient automatic validation of the data.

We have looked into the answer agreement of the workers as an experimental post-processing step. We believe that further work in this area will provide the tools required for automatic data quality control.

#### 5 Conclusions

In this paper we have described a dynamic and inexpensive method of collecting a corpus of questions and answers using the Amazon Mechanical Turk framework. We have provided to the community a corpus of questions, answers and corresponding documents, that we believe can be used in the development of QA-systems for why-questions. We propose that combining several answers from different people is an important factor in defining the “correct” answer to a why-question, and to that goal have included several answers for each question in the corpus.

We have also included data that we believe is valuable in post-processing the data: the work history of a single worker, the time spent on tasks, and the agreement on a single HIT between a set of different workers. We believe that this information, especially the answer agreement of workers, can be successfully used in post-processing and analysing the data, as well as automatically accepting and rejecting workers’ submissions in similar future data collection exercises.

#### Acknowledgments

This study was funded by the Monbusho Scholarship of Japanese Government and the 21st Century COE Program “Framework for Systematization and Application of Large-scale Knowledge Resources (COE-LKR)”

#### References

- Yllias Chali and Maheedhar Kolla. 2004. Summarization Techniques at DUC 2004. In *DUC2004*.
- Hoa Trang Dang, Diane Kelly, and Jimmy Lin. 2007. Overview of the TREC 2007 Question Answering

- Track. In E. Voorhees and L. P. Buckland, editors, *Sixteenth Text REtrieval Conference (TREC)*, Gaithersburg, Maryland, November.
- Ludovic Denoyer and Patrick Gallinari. 2006. The Wikipedia XML Corpus. *SIGIR Forum*.
- Junichi Fukumoto, Tsuneaki Kato, Fumito Masui, and Tsunenori Mori. 2007. An Overview of the 4th Question Answering Challenge (QAC-4) at NTCIR workshop 6. In *Proceedings of the Sixth NTCIR Workshop Meeting*, pages 433–440.
- Chiori Hori, Takaaki Hori, and Sadaoki Furui. 2003. Evaluation Methods for Automatic Speech Summarization. In *In Proc. EUROSPEECH*, volume 4, pages 2825–2828, Geneva, Switzerland.
- Valentin Jijkoun and Maarten de Rijke. 2005. Retrieving Answers from Frequently Asked Questions Pages on the Web. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 76–83, New York, NY, USA. ACM Press.
- Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Human Technology Conference (HLT-NAACL)*, Edmonton, Canada.
- Jimmy Lin and Boris Katz. 2006. Building a Reusable Test Collection for Question Answering. *J. Am. Soc. Inf. Sci. Technol.*, 57(7):851–861.
- Junta Mizuno, Tomoyosi Akiba, Atsushi Fujii, and Katunobu Itou. 2007. Non-factoid Question Answering Experiments at NTCIR-6: Towards Answer Type Detection for Realworld Questions. In *Proceedings of the 6th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, pages 487–492.
- Radu Soricut and Eric Brill. 2006. Automatic Question Answering Using the Web: Beyond the Factoid. *Inf. Retr.*, 9(2):191–206.
- Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2006. Data for Question Answering: the Case of Why. In *LREC*.
- Susan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Discourse-based Answering of Why-questions. *Traitement Automatique des Langues*, 47(2: Discours et document: traitements automatiques):21–41.

# Solving Relational Similarity Problems Using the Web as a Corpus

Preslav Nakov\*

EECS, CS division

University of California at Berkeley  
Berkeley, CA 94720, USA  
nakov@cs.berkeley.edu

Marti A. Hearst

School of Information

University of California at Berkeley  
Berkeley, CA 94720, USA  
hearst@ischool.berkeley.edu

## Abstract

We present a simple linguistically-motivated method for characterizing the semantic relations that hold between two nouns. The approach leverages the vast size of the Web in order to build lexically-specific features. The main idea is to look for verbs, prepositions, and coordinating conjunctions that can help make explicit the hidden relations between the target nouns. Using these features in instance-based classifiers, we demonstrate state-of-the-art results on various relational similarity problems, including mapping noun-modifier pairs to abstract relations like TIME, LOCATION and CONTAINER, characterizing noun-noun compounds in terms of abstract linguistic predicates like CAUSE, USE, and FROM, classifying the relations between nominals in context, and solving SAT verbal analogy problems. In essence, the approach puts together some existing ideas, showing that they apply generally to various semantic tasks, finding that verbs are especially useful features.

## 1 Introduction

Despite the tremendous amount of work on word similarity (see (Budanitsky and Hirst, 2006) for an overview), there is surprisingly little research on the important related problem of *relational similarity* – semantic similarity between pairs of words. Students who took the SAT test before 2005 or who

are taking the GRE test nowadays are familiar with an instance of this problem – verbal analogy questions, which ask whether, e.g., the relationship between *ostrich* and *bird* is more similar to that between *lion* and *cat*, or rather between *primate* and *monkey*. These analogies are difficult, and the average test taker gives a correct answer 57% of the time (Turney and Littman, 2005).

Many NLP applications could benefit from solving relational similarity problems, including but not limited to question answering, information retrieval, machine translation, word sense disambiguation, and information extraction. For example, a relational search engine like TextRunner, which serves queries like “find all X such that X causes wrinkles”, asking for all entities that are in a particular relation with a given entity (Cafarella et al., 2006), needs to recognize that *laugh wrinkles* is an instance of CAUSE-EFFECT. While there are not many success stories so far, measuring semantic similarity has proven its advantages for textual entailment (Tatu and Moldovan, 2005).

In this paper, we introduce a novel linguistically-motivated Web-based approach to relational similarity, which, despite its simplicity, achieves state-of-the-art performance on a number of problems. Following Turney (2006b), we test our approach on SAT verbal analogy questions and on mapping noun-modifier pairs to abstract relations like TIME, LOCATION and CONTAINER. We further apply it to (1) characterizing noun-noun compounds using abstract linguistic predicates like CAUSE, USE, and FROM, and (2) classifying the relation between pairs of nominals in context.

\*After January 2008 at the Linguistic Modeling Department, Institute for Parallel Processing, Bulgarian Academy of Sciences, nakov@lml.bas.bg



## 2 Related Work

### 2.1 Characterizing Semantic Relations

Turney and Littman (2005) characterize the relationship between two words as a vector with coordinates corresponding to the Web frequencies of 128 fixed phrases like “*X for Y*” and “*Y for X*” instantiated from a fixed set of 64 joining terms like *for*, *such as*, *not the*, *is* \*, etc. These vectors are used in a nearest-neighbor classifier to solve SAT verbal analogy problems, yielding 47% accuracy. The same approach is applied to classifying noun-modifier pairs: using the *Diverse* dataset of Nastase and Szpakowicz (2003), Turney&Littman achieve F-measures of 26.5% with 30 fine-grained relations, and 43.2% with 5 course-grained relations.

Turney (2005) extends the above approach by introducing the latent relational analysis (LRA), which uses automatically generated synonyms, learns suitable patterns, and performs singular value decomposition in order to smooth the frequencies. The full algorithm consists of 12 steps described in detail in (Turney, 2006b). When applied to SAT questions, it achieves the state-of-the-art accuracy of 56%. On the *Diverse* dataset, it yields an F-measure of 39.8% with 30 classes, and 58% with 5 classes.

Turney (2006a) presents an unsupervised algorithm for mining the Web for patterns expressing implicit semantic relations. For example, CAUSE (e.g., *cold virus*) is best characterized by “*Y \* causes X*”, and “*Y in \* early X*” is the best pattern for TEMPORAL (e.g., *morning frost*). With 5 classes, he achieves F-measure=50.2%.

### 2.2 Noun-Noun Compound Semantics

Lauer (1995) reduces the problem of noun compound interpretation to choosing the best paraphrasing preposition from the following set: *of*, *for*, *in*, *at*, *on*, *from*, *with* or *about*. He achieved 40% accuracy using corpus frequencies. This result was improved to 55.7% by Lapata and Keller (2005) who used Web-derived *n*-gram frequencies.

Barker and Szpakowicz (1998) use syntactic clues and the identity of the nouns in a nearest-neighbor classifier, achieving 60-70% accuracy.

Rosario and Hearst (2001) used a discriminative classifier to assign 18 relations for noun compounds from biomedical text, achieving 60% accuracy.

Rosario et al. (2002) reported 90% accuracy with a “descent of hierarchy” approach which characterizes the relationship between the nouns in a bio-science noun-noun compound based on the MeSH categories the nouns belong to.

Girju et al. (2005) apply both classic (SVM and decision trees) and novel supervised models (semantic scattering and iterative semantic specialization), using *WordNet*, word sense disambiguation, and a set of linguistic features. They test their system against both Lauer’s 8 prepositional paraphrases and another set of 21 semantic relations, achieving up to 54% accuracy on the latter.

In a previous work (Nakov and Hearst, 2006), we have shown that the relationship between the nouns in a noun-noun compound can be characterized using verbs extracted from the Web, but we provided no formal evaluation.

Kim and Baldwin (2006) characterized the semantic relationship in a noun-noun compound using the verbs connecting the two nouns by comparing them to predefined seed verbs. Their approach is highly resource intensive (uses *WordNet*, *CoreLex* and *Moby’s thesaurus*), and is quite sensitive to the seed set of verbs: on a collection of 453 examples and 19 relations, they achieved 52.6% accuracy with 84 seed verbs, but only 46.7% with 57 seed verbs.

### 2.3 Paraphrase Acquisition

Our method of extraction of paraphrasing verbs and prepositions is similar to previous paraphrase acquisition approaches. Lin and Pantel (2001) extract paraphrases from dependency tree paths whose ends contain semantically similar sets of words by generalizing over these ends. For example, given “*X solves Y*”, they extract paraphrases like “*X finds a solution to Y*”, “*X tries to solve Y*”, “*X resolves Y*”, “*Y is resolved by X*”, etc. The approach is extended by Shinyama et al. (2002), who use named entity recognizers and look for anchors belonging to matching semantic classes, e.g., LOCATION, ORGANIZATION. The idea is further extended by Nakov et al. (2004), who apply it in the biomedical domain, imposing the additional restriction that the sentences from which the paraphrases are extracted cite the same target paper.

## 2.4 Word Similarity

Another important group of related work is on using syntactic dependency features in a vector-space model for measuring *word* similarity, e.g., (Alshawi and Carter, 1994), (Grishman and Sterling, 1994), (Ruge, 1992), and (Lin, 1998). For example, given a noun, Lin (1998) extracts verbs that have that noun as a subject or object, and adjectives that modify it.

## 3 Method

Given a pair of nouns, we try to characterize the semantic relation between them by leveraging the vast size of the Web to build linguistically-motivated lexically-specific features. We mine the Web for sentences containing the target nouns, and we extract the connecting verbs, prepositions, and coordinating conjunctions, which we use in a vector-space model to measure relational similarity.

The process of extraction starts with exact phrase queries issued against a Web search engine (*Google*) using the following patterns:

“*infl*<sub>1</sub> THAT \* *infl*<sub>2</sub>”  
 “*infl*<sub>2</sub> THAT \* *infl*<sub>1</sub>”  
 “*infl*<sub>1</sub> \* *infl*<sub>2</sub>”  
 “*infl*<sub>2</sub> \* *infl*<sub>1</sub>”

where: *infl*<sub>1</sub> and *infl*<sub>2</sub> are inflected variants of *noun*<sub>1</sub> and *noun*<sub>2</sub> generated using the *Java WordNet Library*<sup>1</sup>; THAT is a complementizer and can be *that*, *which*, or *who*; and \* stands for 0 or more (up to 8) instances of *Google*’s star operator.

The first two patterns are subsumed by the last two and are used to obtain more sentences from the search engine since including e.g. *that* in the query changes the set of returned results and their ranking.

For each query, we collect the text snippets from the result set (up to 1,000 per query). We split them into sentences, and we filter out all incomplete ones and those that do not contain the target nouns. We further make sure that the word sequence following the second mentioned target noun is nonempty and contains at least one nonnoun, thus ensuring the snippet includes the entire noun phrase: snippets representing incomplete sentences often end with a period anyway. We then perform POS tagging using the *Stanford POS tagger* (Toutanova et al., 2003)

<sup>1</sup>JWNL: <http://jwordnet.sourceforge.net>

Freq.	Feature	POS	Direction
2205	of	P	2 → 1
1923	be	V	1 → 2
771	include	V	1 → 2
382	serve on	V	2 → 1
189	chair	V	2 → 1
189	have	V	1 → 2
169	consist of	V	1 → 2
148	comprise	V	1 → 2
106	sit on	V	2 → 1
81	be chaired by	V	1 → 2
78	appoint	V	1 → 2
77	on	P	2 → 1
66	and	C	1 → 2
66	be elected	V	1 → 2
58	replace	V	1 → 2
48	lead	V	2 → 1
47	be intended for	V	1 → 2
45	join	V	2 → 1
...	...	...	...
4	be signed up for	V	2 → 1

Table 1: **The most frequent Web-derived features for *committee member*.** Here *V* stands for verb (possibly +preposition and/or +particle), *P* for preposition and *C* for coordinating conjunction; 1 → 2 means *committee* precedes the feature and *member* follows it; 2 → 1 means *member* precedes the feature and *committee* follows it.

and shallow parsing with the *OpenNLP tools*<sup>2</sup>, and we extract the following types of features:

**Verb:** We extract a verb if the subject NP of that verb is headed by one of the target nouns (or an inflected form), and its direct object NP is headed by the other target noun (or an inflected form). For example, the verb *include* will be extracted from “The *committee* includes many *members*.” We also extract verbs from relative clauses, e.g., “This is a *committee* which includes many *members*.” Verb particles are also recognized, e.g., “The *committee* must rotate off 1/3 of its *members*.” We ignore modals and auxiliaries, but retain the passive *be*. Finally, we lemmatize the main verb using *WordNet*’s morphological analyzer *Morphy* (Fellbaum, 1998).

**Verb+Preposition:** If the subject NP of a verb is headed by one of the target nouns (or an inflected form), and its indirect object is a PP containing an NP which is headed by the other target noun (or an inflected form), we extract the verb and the preposi-

<sup>2</sup>OpenNLP: <http://opennlp.sourceforge.net>

tion heading that PP, e.g., “The thesis advisory *committee* consists of three qualified *members*.” As in the verb case, we extract verb+preposition from relative clauses, we include particles, we ignore modals and auxiliaries, and we lemmatize the verbs.

**Preposition:** If one of the target nouns is the head of an NP containing a PP with an internal NP headed by the other target noun (or an inflected form), we extract the preposition heading that PP, e.g., “The *members of the committee* held a meeting.”

**Coordinating conjunction:** If the two target nouns are the heads of coordinated NPs, we extract the coordinating conjunction.

In addition to the lexical part, for each extracted feature, we keep a direction. Therefore the preposition *of* represents two different features in the following examples “*member of the committee*” and “*committee of members*”. See Table 1 for examples.

We use the above-described features to calculate relational similarity, i.e., similarity between pairs of nouns. In order to downweight very common features like *of*, we use TF.IDF-weighting:

$$w(x) = TF(x) \times \log \left( \frac{N}{DF(x)} \right) \quad (1)$$

In the above formula,  $TF(x)$  is the number of times the feature  $x$  has been extracted for the target noun pair,  $DF(x)$  is the total number of training noun pairs that have that feature, and  $N$  is the total number of training noun pairs.

Given two nouns and their TF.IDF-weighted frequency vectors  $A$  and  $B$ , we calculate the similarity between them using the following generalized variant of the Dice coefficient:

$$Dice(A, B) = \frac{2 \times \sum_{i=1}^n \min(a_i, b_i)}{\sum_{i=1}^n a_i + \sum_{i=1}^n b_i} \quad (2)$$

Other variants are also possible, e.g., Lin (1998).

## 4 Relational Similarity Experiments

### 4.1 SAT Verbal Analogy

Following Turney (2006b), we use *SAT verbal analogy* as a benchmark problem for relational similarity. We experiment with the 374 SAT questions collected by Turney and Littman (2005). Table 2 shows two sample questions: the top word pairs

<b>ostrich:bird</b>		<b>palatable:toothsome</b>	
(a)	<i>lion:cat</i>	(a)	rancid:fragrant
(b)	goose:flock	(b)	chewy:textured
(c)	ewe:sheep	(c)	<i>coarse:rough</i>
(d)	cub:bear	(d)	solitude:company
(e)	primate:monkey	(e)	no choice

Table 2: **SAT verbal analogy: sample questions.** The stem is in **bold**, the correct answer is in *italic*, and the distractors are in plain text.

are called *stems*, the ones in italic are the *solutions*, and the remaining ones are *distractors*. Turney (2006b) achieves 56% accuracy on this dataset, which matches the average human performance of 57%, and represents a significant improvement over the 20% random-guessing baseline.

Note that the righthand side example in Table 2 is missing one distractor; so do 21 questions. The dataset also mixes different parts of speech: while *solitude* and *company* are nouns, all remaining words are adjectives. Other examples contain verbs and adverbs, and even relate pairs of different POS. This is problematic for our approach, which requires that both words be nouns<sup>3</sup>. After having filtered all examples containing nonnouns, we ended up with 184 questions, which we used in the evaluation.

Given a verbal analogy example, we build six feature vectors – one for each of the six word pairs. We then calculate the relational similarity between the stem of the analogy and each of the five candidates, and we choose the pair with the highest score; we make no prediction in case of a tie.

The evaluation results for a leave-one-out cross-validation are shown in Table 3. We also show 95%-confidence intervals for the accuracy. The last line in the table shows the performance of Turney’s LRA when limited to the 184 noun-only examples. Our best model  $v + p + c$  performs a bit better, 71.3% vs. 67.4%, but the difference is not statistically significant. However, this “inferred” accuracy could be misleading, and the LRA could have performed better if it was restricted to solve *noun-only* analogies, which seem easier than the general ones, as demonstrated by the significant increase in accuracy for LRA when limited to nouns: 67.4% vs. 56%.

<sup>3</sup>It can be extended to handle adjective-noun pairs as well, as demonstrated in section 4.2 below.

Model	✓	×	∅	Accuracy	Cover.
$v + p + c$	129	52	3	<b>71.3±7.0</b>	<b>98.4</b>
$v$	122	56	6	68.5±7.2	96.7
$v + p$	119	61	4	66.1±7.2	97.8
$v + c$	117	62	5	65.4±7.2	97.3
$p + c$	90	90	4	50.0±7.2	97.8
$p$	84	94	6	47.2±7.2	96.7
baseline	37	147	0	20.0±5.2	100.0
LRA	122	59	3	67.4±7.1	98.4

Table 3: **SAT verbal analogy: 184 noun-only examples.**  $v$  stands for verb,  $p$  for preposition, and  $c$  for coordinating conjunction. For each model, the number of correct (✓), wrong (×), and nonclassified examples (∅) is shown, followed by accuracy and coverage (in %s).

Model	✓	×	∅	Accuracy	Cover.
$v + p$	240	352	8	<b>40.5±3.9</b>	<b>98.7</b>
$v + p + c$	238	354	8	40.2±3.9	98.7
$v$	234	350	16	40.1±3.9	97.3
$v + c$	230	362	8	38.9±3.8	98.7
$p + c$	114	471	15	19.5±3.0	97.5
$p$	110	475	15	19.1±3.0	97.5
baseline	49	551	0	8.2±1.9	100.0
LRA	239	361	0	39.8±3.8	100.0

Table 4: **Head-modifier relations, 30 classes:** evaluation on the *Diverse* dataset, micro-averaged (in %s).

## 4.2 Head-Modifier Relations

Next, we experiment with the *Diverse* dataset of Barker and Szpakowicz (1998), which consists of 600 head-modifier pairs: noun-noun, adjective-noun and adverb-noun. Each example is annotated with one of 30 fine-grained relations, which are further grouped into the following 5 coarse-grained classes (the fine-grained relations are shown in parentheses): CAUSALITY (*cause, effect, purpose, detraction*), TEMPORALITY (*frequency, time\_at, time\_through*), SPATIAL (*direction, location, location\_at, location\_from*), PARTICIPANT (*agent, beneficiary, instrument, object, object\_property, part, possessor, property, product, source, stative, whole*) and QUALITY (*container, content, equative, material, measure, topic, type*). For example, *exam anxiety* is classified as *effect* and therefore as CAUSALITY, and *blue book* is *property* and therefore also PARTICIPANT.

Some examples in the dataset are problematic for our method. First, in three cases, there are two mod-

ifiers, e.g., *infectious disease agent*, and we had to ignore the first one. Second, seven examples have an adverb modifier, e.g., *daily exercise*, and 262 examples have an adjective modifier, e.g., *tiny cloud*. We treat them as if the modifier was a noun, which works in many cases, since many adjectives and adverbs can be used predicatively, e.g., ‘*This exercise is performed daily.*’ or ‘*This cloud looks very tiny.*’

For the evaluation, we created a feature vector for each head-modifier pair, and we performed a leave-one-out cross-validation: we left one example for testing and we trained on the remaining 599 ones, repeating this procedure 600 times so that each example be used for testing. Following Turney and Littman (2005) we used a 1-nearest-neighbor classifier. We calculated the similarity between the feature vector of the testing example and each of the training examples’ vectors. If there was a unique most similar training example, we predicted its class, and if there were ties, we chose the class predicted by the majority of tied examples, if there was a majority.

The results for the 30-class *Diverse* dataset are shown in Table 4. Our best model achieves 40.5% accuracy, which is slightly better than LRA’s 39.8%, but the difference is not statistically significant.

Table 4 shows that the verbs are the most important features, yielding about 40% accuracy regardless of whether used alone or in combination with prepositions and/or coordinating conjunctions; not using them results in 50% drop in accuracy.

The reason coordinating conjunctions do not help is that head-modifier relations are typically expressed with verbal or prepositional paraphrases. Therefore, coordinating conjunctions only help with some infrequent relations like *equative*, e.g., *finding player and coach* on the Web suggests an equative relation for *player coach* (and for *coach player*).

As Table 3 shows, this is different for SAT verbal analogy, where verbs are still the most important feature type and the only whose presence/absence makes a statistical difference. However, this time coordinating conjunctions (with prepositions) do help a bit (the difference is not statistically significant) since SAT verbal analogy questions ask for a broader range of relations, e.g., antonymy, for which coordinating conjunctions like *but* are helpful.

Model	Accuracy
$v + p + c + sent + query$ (type <i>C</i> )	<b>68.1±4.0</b>
$v$	67.9±4.0
$v + p + c$	67.8±4.0
$v + p + c + sent$ (type <i>A</i> )	<b>67.3±4.0</b>
$v + p$	66.9±4.0
$sent$ (sentence words only)	59.3±4.2
$p$	58.4±4.2
Baseline (majority class)	57.0±4.2
$v + p + c + sent + query$ ( <i>C</i> ), 8 stars	67.0±4.0
$v + p + c + sent$ ( <i>A</i> ), 8 stars	65.4±4.1
Best type <i>C</i> on <i>SemEval</i>	67.0±4.0
Best type <i>A</i> on <i>SemEval</i>	66.0±4.1

Table 5: **Relations between nominals:** evaluation on the *SemEval* dataset. Accuracy is macro-averaged (in %s), up to 10 *Google* stars are used unless otherwise stated.

### 4.3 Relations Between Nominals

We further experimented with the *SemEval'07* task 4 dataset (Girju et al., 2007), where each example consists of a sentence, a target semantic relation, two nominals to be judged on whether they are in that relation, manually annotated *WordNet* senses, and the Web query used to obtain the sentence:

```
"Among the contents of the
<e1>vessel</e1> were a set of
carpenter's <e2>tools</e2>, several
large storage jars, ceramic utensils,
ropes and remnants of food, as well
as a heavy load of ballast stones."
WordNet(e1) = "vessel%1:06:00::",
WordNet(e2) = "tool%1:06:00::",
Content-Container(e2, e1) = "true",
Query = "contents of the * were a"
```

The following nonexhaustive and possibly overlapping relations are possible: Cause-Effect (e.g., *hormone-growth*), Instrument-Agency (e.g., *laser-printer*), Theme-Tool (e.g., *workforce*), Origin-Entity (e.g., *grain-alcohol*), Content-Container (e.g., *bananas-basket*), Product-Producer (e.g., *honey-bee*), and Part-Whole (e.g., *leg-table*). Each relation is considered in isolation; there are 140 training and at least 70 test examples per relation.

Given an example, we reduced the target entities  $e_1$  and  $e_2$  to single nouns by retaining their heads only. We then mined the Web for sentences con-

taining these nouns, and we extracted the above-described feature types: verbs, prepositions and coordinating conjunctions. We further used the following problem-specific contextual feature types:

**Sentence words:** after stop words removal and stemming with the Porter (1980) stemmer;

**Entity words:** lemmata of the words in  $e_1$  and  $e_2$ ;

**Query words:** words part of the query string.

Each feature type has a specific prefix which prevents it from mixing with other feature types; the last feature type is used for type *C* only (see below).

The *SemEval* competition defines four types of systems, depending on whether the manually annotated *WordNet* senses and the *Google* query are used: *A* (WordNet=no, Query=no), *B* (WordNet=yes, Query=no), *C* (WordNet=no, Query=yes), and *D* (WordNet=yes, Query=yes). We experimented with types *A* and *C* only since we believe that having the manually annotated *WordNet* sense keys is an unrealistic assumption for a real-world application.

As before, we used a 1-nearest-neighbor classifier with TF.IDF-weighting, breaking ties by predicting the majority class on the training data. The evaluation results are shown in Table 5. We studied the effect of different subsets of features and of more *Google* star operators. As the table shows, using up to ten *Google* stars instead of up to eight (see section 3) yields a slight improvement in accuracy for systems of both type *A* (65.4% vs. 67.3%) and type *C* (67.0% vs. 68.1%). Both results represent a statistically significant improvement over the majority class baseline and over using sentence words only, and a slight improvement over the best type *A* and type *C* systems on *SemEval'07*, which achieved 66% and 67% accuracy, respectively.<sup>4</sup>

### 4.4 Noun-Noun Compound Relations

The last dataset we experimented with is a subset of the 387 examples listed in the appendix of (Levi, 1978). Levi's theory is one of the most important linguistic theories of the syntax and semantics of *complex nominals* – a general concept grouping

<sup>4</sup>The best type *B* system on *SemEval* achieved 76.3% accuracy using the manually-annotated *WordNet* senses in context for each example, which constitutes an additional data source, as opposed to an additional resource. The systems that used *WordNet* as a resource only, i.e., ignoring the manually annotated senses, were classified as type *A* or *C*. (Girju et al., 2007)

Model	USING THAT				NOT USING THAT			
	Accuracy	Cover.	ANF	ASF	Accuracy	Cover.	ANF	ASF
Human: all $v$	78.4±6.0	99.5	34.3	70.9	–	–	–	–
Human: first $v$ from each worker	72.3±6.4	99.5	11.6	25.5	–	–	–	–
$v + p + c$	50.0±6.7	99.1	216.6	1716.0	49.1±6.7	99.1	206.6	1647.6
$v + p$	50.0±6.7	99.1	208.9	1427.9	47.6±6.6	99.1	198.9	1359.5
$v + c$	46.7±6.6	99.1	187.8	1107.2	43.9±6.5	99.1	177.8	1038.8
$v$	45.8±6.6	99.1	180.0	819.1	42.9±6.5	99.1	170.0	750.7
$p$	33.0±6.0	99.1	28.9	608.8	33.0±6.0	99.1	28.9	608.8
$p + c$	32.1±5.9	99.1	36.6	896.9	32.1±5.9	99.1	36.6	896.9
Baseline	19.6±4.8	100.0	–	–	–	–	–	–

Table 6: **Noun-noun compound relations, 12 classes:** evaluation on *Levi-214* dataset. Shown are micro-averaged accuracy and coverage in %s, followed by average number of features (ANF) and average sum of feature frequencies (ASF) per example. The righthand side reports the results when the query patterns involving THAT were not used. For comparison purposes, the top rows show the performance with the human-proposed verbs used as features.

together the partially overlapping classes of nominal compounds (e.g., *peanut butter*), nominalizations (e.g., *dream analysis*), and nonpredicate noun phrases (e.g., *electric shock*).

In Levi’s theory, complex nominals can be derived from relative clauses by removing one of the following 12 abstract predicates: CAUSE<sub>1</sub> (e.g., *tear gas*), CAUSE<sub>2</sub> (e.g., *drug deaths*), HAVE<sub>1</sub> (e.g., *apple cake*), HAVE<sub>2</sub> (e.g., *lemon peel*), MAKE<sub>1</sub> (e.g., *silkworm*), MAKE<sub>2</sub> (e.g., *snowball*), USE (e.g., *steam iron*), BE (e.g., *soldier ant*), IN (e.g., *field mouse*), FOR (e.g., *horse doctor*), FROM (e.g., *olive oil*), and ABOUT (e.g., *price war*). In the resulting nominals, the modifier is typically the object of the predicate; when it is the subject, the predicate is marked with the index 2. The second derivational mechanism in the theory is nominalization; it produces nominals whose head is a nominalized verb.

Since we are interested in noun compounds only, we manually cleansed the set of 387 examples. We first excluded all concatenations (e.g., *silkworm*) and examples with adjectival modifiers (e.g., *electric shock*), thus obtaining 250 noun-noun compounds (*Levi-250* dataset). We further filtered out all nominalizations for which the dataset provides no abstract predicate (e.g., *city planner*), thus ending up with 214 examples (*Levi-214* dataset).

As in the previous experiments, for each of the 214 noun-noun compounds, we mined the Web for sentences containing both target nouns, from which we extracted paraphrasing verbs, prepositions

and coordinating conjunctions. We then performed leave-one-out cross-validation experiments with a 1-nearest-neighbor classifier, trying to predict the correct predicate for the testing example. The results are shown in Table 6. As we can see, using prepositions alone yields about 33% accuracy, which is a statistically significant improvement over the majority-class baseline. Overall, the most important features are the verbs: they yield 45.8% accuracy when used alone, and 50% together with prepositions. Adding coordinating conjunctions helps a bit with verbs, but not with prepositions. Note however that none of the differences between the different feature combinations involving verbs are statistically significant.

The righthand side of the table reports the results when the query patterns involving THAT (see section 3) were not used. We can observe a small 1-3% drop in accuracy for all models involving verbs, but it is not statistically significant.

We also show the average number of distinct features and sum of feature counts per example: as we can see, there is a strong positive correlation between number of features and accuracy.

## 5 Comparison to Human Judgments

Since in all above tasks the most important features were the verbs, we decided to compare our Web-derived verbs to human-proposed ones for all noun-noun compounds in the *Levi-250* dataset. We asked human subjects to produce verbs, possibly

followed by prepositions, that could be used in a paraphrase involving *that*. For example, *olive oil* can be paraphrased as ‘*oil that comes from olives*’, ‘*oil that is obtained from olives*’ or ‘*oil that is from olives*’. Note that this implicitly allows for prepositional paraphrases – when the verb is to *be* and is followed by a preposition, as in the last paraphrase.

We used the *Amazon Mechanical Turk* Web service<sup>5</sup> to recruit human subjects, and we instructed them to propose at least three paraphrasing verbs per noun-noun compound, if possible. We randomly distributed the noun-noun compounds into groups of 5 and we requested 25 different human subjects per group. Each human subject was allowed to work on any number of groups, but not on the same one twice. A total of 174 different human subjects produced 19,018 verbs. After filtering the bad submissions and normalizing the verbs, we ended up with 17,821 verbs. See (Nakov, 2007) for further details on the process of extraction and cleansing. The dataset itself is freely available (Nakov, 2008).

We compared the human-proposed and the Web-derived verbs for *Levi-214*, aggregated by relation. Given a relation, we collected all verbs belonging to noun-noun compounds from that relation together with their frequencies. From a vector-space model point of view, we summed their corresponding frequency vectors. We did this separately for the human- and the program-generated verbs, and we compared the resulting vectors using Dice coefficient with TF.IDF, calculated as before. Figure 1 shows the cosine correlations using all human-proposed verbs and the first verb from each judge. We can see a very-high correlation (mid-70% to mid-90%) for relations like CAUSE<sub>1</sub>, MAKE<sub>1</sub>, BE, but low correlations of 11-30% for reverse relations like HAVE<sub>2</sub> and MAKE<sub>2</sub>. Interestingly, using the first verb only improves the results for highly-correlated relations, but negatively affects low-correlated ones.

Finally, we repeated the cross-validation experiment with the *Levi-214* dataset, this time using the human-proposed verbs<sup>6</sup> as features. As Table 6 shows, we achieved 78.4% accuracy using all verbs (and and 72.3% with the first verb from each worker), which is a statistically significant improve-

<sup>5</sup><http://www.mturk.com>

<sup>6</sup>Note that the human subjects proposed their verbs without any context and independently of our Web-derived sentences.

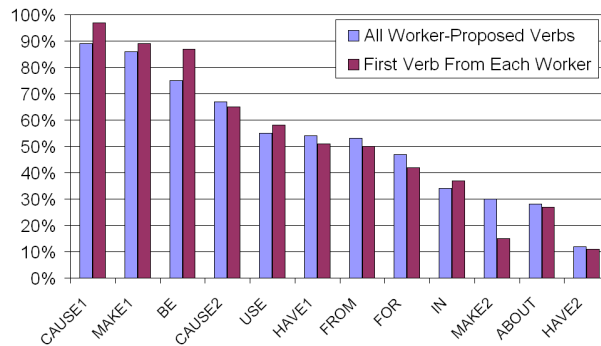


Figure 1: **Cosine correlation (in %s) between the human- and the program- generated verbs by relation:** using all human-proposed verbs vs. the first verb.

ment over the 50% of our best Web-based model. This result is strong for a 12-way classification problem, and confirms our observation that verbs and prepositions are among the most important features for relational similarity problems. It further suggests that the human-proposed verbs might be an upper bound on the accuracy that could be achieved with automatically extracted features.

## 6 Conclusions and Future Work

We have presented a simple approach for characterizing the relation between a pair of nouns in terms of linguistically-motivated features which could be useful for many NLP tasks. We found that verbs were especially useful features for this task. An important advantage of the approach is that it does not require knowledge about the semantics of the individual nouns. A potential drawback is that it might not work well for low-frequency words.

The evaluation on several relational similarity problems, including SAT verbal analogy, head-modifier relations, and relations between complex nominals has shown state-of-the-art performance. The presented approach can be further extended to other combinations of parts of speech: not just noun-noun and adjective-noun. Using a parser with a richer set of syntactic dependency features, e.g., as proposed by Padó and Lapata (2007), is another promising direction for future work.

## Acknowledgments

This research was supported in part by NSF DBI-0317510.

## References

- Hiyan Alshawi and David Carter. 1994. Training and scaling preference functions for disambiguation. *Computational Linguistics*, 20(4):635–648.
- Ken Barker and Stan Szpakowicz. 1998. Semi-automatic recognition of noun modifier relationships. In *Proc. of Computational linguistics*, pages 96–102.
- Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.
- Michael Cafarella, Michele Banko, and Oren Etzioni. 2006. Relational Web search. Technical Report 2006-04-02, University of Washington, Department of Computer Science and Engineering.
- Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.
- 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*, 4(19):479–496.
- Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of SemEval*, pages 13–18, Prague, Czech Republic.
- Ralph Grishman and John Sterling. 1994. Generalizing automatically generated selectional patterns. In *Proceedings of the 15th conference on Computational linguistics*, pages 742–747.
- Su Nam Kim and Timothy Baldwin. 2006. Interpreting semantic relations in noun compounds via verb semantics. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 491–498.
- Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Trans. Speech Lang. Process.*, 2(1):3.
- Mark Lauer. 1995. *Designing Statistical Language Learners: Experiments on Noun Compounds*. Ph.D. thesis, Dept. of Computing, Macquarie University, Australia.
- Judith Levi. 1978. *The Syntax and Semantics of Complex Nominals*. Academic Press, New York.
- Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360.
- Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of ICML*, pages 296–304.
- Preslav Nakov and Marti Hearst. 2006. Using verbs to characterize noun-noun relations. In *AIMSA*, volume 4183 of *LNCS*, pages 233–244. Springer.
- Preslav Nakov, Ariel Schwartz, and Marti Hearst. 2004. Citances: Citation sentences for semantic analysis of bioscience text. In *Proceedings of SIGIR'04 Workshop on Search and Discovery in Bioinformatics*, pages 81–88, Sheffield, UK.
- Preslav Nakov. 2007. *Using the Web as an Implicit Training Set: Application to Noun Compound Syntax and Semantics*. Ph.D. thesis, EECS Department, University of California, Berkeley, UCB/EECS-2007-173.
- Preslav Nakov. 2008. Paraphrasing verbs for noun compound interpretation. In *Proceedings of the LREC'08 Workshop: Towards a Shared Task for Multiword Expressions (MWE'08)*, Marrakech, Morocco.
- Vivi Nastase and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 285–301, Tilburg, The Netherlands.
- Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.
- Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.
- Barbara Rosario and Marti Hearst. 2001. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proceedings of EMNLP*, pages 82–90.
- Barbara Rosario, Marti Hearst, and Charles Fillmore. 2002. The descent of hierarchy, and selection in relational semantics. In *Proceedings of ACL*, pages 247–254.
- Gerda Ruge. 1992. Experiment on linguistically-based term associations. *Inf. Process. Manage.*, 28(3):317–332.
- Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of HLT*, pages 313–318.
- Marta Tatu and Dan Moldovan. 2005. A semantic approach to recognizing textual entailment. In *Proceedings of HLT*, pages 371–378.
- Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, pages 252–259.
- Peter Turney and Michael Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning Journal*, 60(1-3):251–278.
- Peter Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proceedings of IJCAI*, pages 1136–1141.
- Peter Turney. 2006a. Expressing implicit semantic relations without supervision. In *Proceedings of ACL*, pages 313–320.
- Peter Turney. 2006b. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.



# Combining Speech Retrieval Results with Generalized Additive Models

J. Scott Olsson\* and Douglas W. Oard†

UMIACS Laboratory for Computational Linguistics and Information Processing  
University of Maryland, College Park, MD 20742

Human Language Technology Center of Excellence  
John Hopkins University, Baltimore, MD 21211  
olsson@math.umd.edu, oard@umd.edu

## Abstract

Rapid and inexpensive techniques for automatic transcription of speech have the potential to dramatically expand the types of content to which information retrieval techniques can be productively applied, but limitations in accuracy and robustness must be overcome before that promise can be fully realized. Combining retrieval results from systems built on various errorful representations of the same collection offers some potential to address these challenges. This paper explores that potential by applying Generalized Additive Models to optimize the combination of ranked retrieval results obtained using transcripts produced automatically for the same spoken content by substantially different recognition systems. Topic-averaged retrieval effectiveness better than any previously reported for the same collection was obtained, and even larger gains are apparent when using an alternative measure emphasizing results on the most difficult topics.

## 1 Introduction

Speech retrieval, like other tasks that require transforming the representation of language, suffers from both random and systematic errors that are introduced by the speech-to-text transducer. Limitations in signal processing, acoustic modeling, pronunciation, vocabulary, and language modeling can be accommodated in several ways, each of which make different trade-offs and thus induce different

error characteristics. Moreover, different applications produce different types of challenges and different opportunities. As a result, optimizing a single recognition system for all transcription tasks is well beyond the reach of present technology, and even systems that are apparently similar on average can make different mistakes on different sources. A natural response to this challenge is to combine retrieval results from multiple systems, each imperfect, to achieve reasonably robust behavior over a broader range of tasks. In this paper, we compare alternative ways of combining these ranked lists. Note, we do not assume access to the internal workings of the recognition systems, or even to the transcripts produced by those systems.

System combination has a long history in information retrieval. Most often, the goal is to combine results from systems that search different content (“collection fusion”) or to combine results from different systems on the same content (“data fusion”). When working with multiple *transcriptions* of the same content, we are again presented with new opportunities. In this paper we compare some well known techniques for combination of retrieval results with a new evidence combination technique based on a general framework known as *Generalized Additive Models* (GAMs). We show that this new technique significantly outperforms several well known information retrieval fusion techniques, and we present evidence that it is the ability of GAMs to combine inputs non-linearly that at least partly explains our improvements.

The remainder of this paper is organized as follows. We first review prior work on evidence com-

\* Dept. of Mathematics/AMSC, UMD

† College of Information Studies, UMD

bination in information retrieval in Section 2, and then introduce Generalized Additive Models in Section 3. Section 4 describes the design of our experiments with a 589 hour collection of conversational speech for which information retrieval queries and relevance judgments are available. Section 5 presents the results of our experiments, and we conclude in Section 6 with a brief discussion of implications of our results and the potential for future work on this important problem.

## 2 Previous Work

One approach for combining ranked retrieval results is to simply linearly combine the multiple system scores for each topic and document. This approach has been extensively applied in the literature (Bartell et al., 1994; Callan et al., 1995; Powell et al., 2000; Vogt and Cottrell, 1999), with varying degrees of success, owing in part to the potential difficulty of normalizing scores across retrieval systems. In this study, we partially abstract away from this potential difficulty by using the same retrieval system on both representations of the collection documents (so that we don't expect score distributions to be significantly different for the combination inputs).

Of course, many fusion techniques using more advanced score normalization methods have been proposed. Shaw and Fox (1994) proposed a number of such techniques, perhaps the most successful of which is known as CombMNZ. CombMNZ has been shown to achieve strong performance and has been used in many subsequent studies (Lee, 1997; Montague and Aslam, 2002; Beitzel et al., 2004; Lillis et al., 2006). In this study, we also use CombMNZ as a baseline for comparison, and following Lillis et al. (2006) and Lee (1997), compute it in the following way. First, we normalize each score  $s_i$  as  $norm(s_i) = \frac{s_i - min(s)}{max(s) - min(s)}$ , where  $max(s)$  and  $min(s)$  are the maximum and minimum scores seen in the input result list. After normalization, the CombMNZ score for a document  $d$  is computed as

$$CombMNZ_d = \sum_{\ell} N_{s,d} \times |N_d > 0|.$$

Here,  $\mathcal{L}$  is the number of ranked lists to be combined,  $N_{\ell,d}$  is the normalized score of document  $d$

in ranked list  $\ell$ , and  $|N_d > 0|$  is the number of non-zero normalized scores given to  $d$  by any result set.

Manmatha et al. (2001) showed that retrieval scores from IR systems could be modeled using a Normal distribution for relevant documents and exponential distribution for non-relevant documents. However, in their study, fusion results using these comparatively complex normalization approaches achieved performance no better than the much simpler CombMNZ.

A simple rank-based fusion technique is *interleaving* (Voorhees et al., 1994). In this approach, the highest ranked document from each list is taken in turn (ignoring duplicates) and placed at the top of the new, combined list.

Many probabilistic combination approaches have also been developed, a recent example being Lillis et al. (2006). Perhaps the most closely related proposal, using logistic regression, was made first by Savoy et al. (1988). Logistic regression is one example from the broad class of models which GAMs encompass. Unlike GAMs in their full generality however, logistic regression imposes a comparatively high degree of linearity in the model structure.

### 2.1 Combining speech retrieval results

Previous work on single-collection result fusion has naturally focused on combining results from multiple retrieval systems. In this case, the potential for performance improvements depends critically on the uniqueness of the different input systems being combined. Accordingly, small variations in the same system often do not combine to produce results better than the best of their inputs (Beitzel et al., 2004).

Errorful document collections such as conversational speech introduce new difficulties and opportunities for data fusion. This is so, in particular, because even the same system can produce drastically different retrieval results when multiple representations of the documents (e.g., multiple transcript hypotheses) are available. Consider, for example, Figure 1 which shows, for each term in each of our title queries, the proportion of relevant documents containing that term *in only one* of our two transcript hypotheses. Critically, by plotting this proportion against the term's inverse document frequency, we observe that the most discriminative query terms are often not available in both document represen-

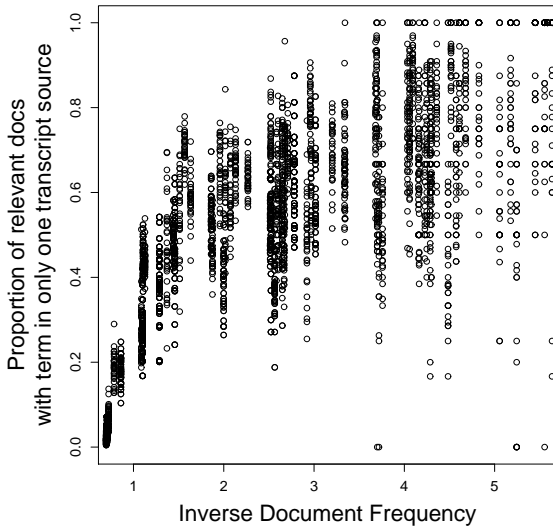


Figure 1: For each term in each query, the proportion of relevant documents containing the term vs. inverse document frequency. For increasingly discriminative terms (higher *idf*), we observe that the probability of only one transcript containing the term increases dramatically.

tations. As these high-*idf* terms make large contributions to retrieval scores, this suggests that even an identical retrieval system may return a large score using one transcript hypothesis, and yet a very low score using another. Accordingly, a linear combination of scores is unlikely to be optimal.

A second example illustrates the difficulty. Suppose recognition system *A* can recognize a particular high-*idf* query term, but system *B* never can. In the extreme case, the term may simply be out of vocabulary, although this may occur for various other reasons (e.g., poor language modeling or pronunciation dictionaries). Here again, a linear combination of scores will fail, as will rank-based interleaving. In the latter case, we will alternate between taking a plausible document from system *A* and an inevitably worse result from the crippled system *B*.

As a potential solution for these difficulties, we consider the use of generalized additive models for retrieval fusion.

### 3 Generalized Additive Models

*Generalized Additive Models* (GAMs) are a generalization of *Generalized Linear Models* (GLMs),

while GLMs are a generalization of the well known linear model. In a GLM, the distribution of an observed random variable  $Y_i$  is related to the linear predictor  $\eta_i$  through a smooth monotonic *link function*  $g$ ,

$$g(\mu_i) = \eta_i = \mathbf{X}_i\beta.$$

Here,  $\mathbf{X}_i$  is the  $i^{\text{th}}$  row of the model matrix  $\mathbf{X}$  (one set of observations corresponding to one observed  $y_i$ ) and  $\beta$  is a vector of unknown parameters to be learned from the data. If we constrain our link function  $g$  to be the identity transformation, and assume  $Y_i$  is Normal, then our GLM reduces to a simple linear model.

But GLMs are considerably more versatile than linear models. First, rather than only the Normal distribution, the response  $Y_i$  is free to have any distribution belonging to the exponential family of distributions such as the Binomial, Normal, Gamma, and Poisson. Secondly, by allowing non-identity link functions  $g$ , some degree of non-linearity may be incorporated in the model structure.

A well known GLM in the NLP community is *logistic regression* (which may alternatively be derived as a maximum entropy classifier). In logistic regression, the response is assumed to be Binomial and the chosen link function is the logit transformation,

$$g(\mu_i) = \text{logit}(\mu_i) = \log\left(\frac{\mu_i}{1 - \mu_i}\right)$$

Generalized additive models allow for additional model flexibility by allowing the linear predictor to now also contain learned smooth functions  $f_j$  of the covariates  $x_k$ . For example,

$$g(\mu_i) = \mathbf{X}_i^*\theta + f_1(x_{1i}) + f_2(x_{2i}) + f_3(x_{3i}, x_{4i}).$$

As in a GLM,  $\mu_i \equiv \mathbb{E}(Y_i)$  and  $Y_i$  belongs to the exponential family. Strictly parametric model components are still permitted, which we represent as a row of the model matrix  $\mathbf{X}_i^*$  (with associated parameters  $\theta$ ).

GAMs may be thought of as GLMs where one or more covariate has been transformed by a basis expansion,  $f(x) = \sum_j^q b_j(x)\beta_j$ . Given a set of  $q$  basis functions  $b_j$  spanning a  $q$ -dimensional space

of smooth transformations, we are back to the linear problem of learning coefficients  $\beta_j$  which “optimally” fit the data. If we knew the appropriate transformation of our covariates (say the logarithm), we could simply apply it ourselves. GAMs allow us to learn these transformations from the data, when we expect some transformation to be useful but don’t know its form *a priori*. In practice, these smooth functions may be represented and the model parameters may be learned in various ways. In this work, we use the excellent open source package `mgcv` (Wood, 2006), which uses penalized likelihood maximization to prevent arbitrarily “wiggly” smooth functions (i.e., overfitting). Smooths (including multidimensional smooths) are represented by thin plate regression splines (Wood, 2003).

### 3.1 Combining speech retrieval results with GAMs

The chief difficulty introduced in combining ranked speech retrieval results is the severe disagreement introduced by differing document hypotheses. As we saw in Figure 1, it is often the case that the most discriminative query terms occur in only one transcript source.

#### 3.1.1 GLM with factors

Our first new approach for handling differences in transcripts is an extension of the logistic regression model previously used in data fusion work, (Savoy et al., 1988). Specifically, we augment the model with the first-order interaction of scores  $x_1x_2$  and the factor  $\alpha_i$ , so that

$$\text{logit}\{\mathbb{E}(R_i)\} = \beta_0 + \alpha_i + x_1\beta_1 + x_2\beta_2 + x_1x_2\beta_3,$$

where the relevance  $R_i \sim \text{Binomial}$ . A factor is essentially a learned intercept for different subsets of the response. In this case,

$$\alpha_i = \begin{cases} \beta_{BOTH} & \text{if both representations matched } q_i \\ \beta_{IBM} & \text{only } d_{i,IBM} \text{ matched } q_i \\ \beta_{BBN} & \text{only } d_{i,BBN} \text{ matched } q_i \end{cases}$$

where  $\alpha_i$  corresponds to data row  $i$ , with associated document representations  $d_{i,source}$  and query  $q_i$ . The intuition is simply that we’d like our model to have different biases for or against relevance

based on which transcript source retrieved the document. This is a small-dimensional way of dampening the effects of significant disagreements in the document representations.

#### 3.1.2 GAM with multidimensional smooth

If a document’s score is large in both systems, we expect it to have high probability of relevance. However, as a document’s score increases linearly in one source, we have no reason to expect its probability of relevance to also increase linearly. Moreover, because the most discriminative terms are likely to be found in only one transcript source, even an absent score for a document does not ensure a document is not relevant. It is clear then that the mapping from document scores to probability of relevance is in general a complex nonlinear surface. The limited degree of nonlinear structure afforded to GLMs by non-identity link functions is unlikely to sufficiently capture this intuition.

Instead, we can model this non-linearity using a generalized additive model with multidimensional smooth  $f(x_{IBM}, x_{BBN})$ , so that

$$\text{logit}\{\mathbb{E}(R_i)\} = \beta_0 + f(x_{IBM}, x_{BBN}).$$

Again,  $R_i \sim \text{Binomial}$  and  $\beta_0$  is a learned intercept (which, alternatively, may be absorbed by the smooth  $f$ ).

Figure 2 shows the smoothing transformation  $f$  learned during our evaluation. Note the small decrease in predicted probability of relevance as the retrieval score from one system decreases, while the probability curves upward again as the disagreement increases. This captures our intuition that systems often disagree strongly because discriminative terms are often not recognized in all transcript sources.

We can think of the probability of relevance mapping learned by the factor model of Section 3.1.1 as also being a surface defined over the space of input document scores. That model, however, was constrained to be linear. It may be visualized as a collection of affine planes (with common normal vectors, but each shifted upwards by their factor level’s weight and the common intercept).

## 4 Experiments

### 4.1 Dataset

Our dataset is a collection of 272 oral history interviews from the MALACH collection. The task is to retrieve short speech segments which were manually designated as being topically coherent by professional indexers. There are 8,104 such segments (corresponding to roughly 589 hours of conversational speech) and 96 assessed topics. We follow the topic partition used for the 2007 evaluation by the Cross Language Evaluation Forum’s cross-language speech retrieval track (Pecina et al., 2007). This gives us 63 topics on which to train our combination systems and 33 topics for evaluation.

### 4.2 Evaluation

#### 4.2.1 Geometric Mean Average Precision

Average precision (AP) is the average of the precision values obtained after each document relevant to a particular query is retrieved. To assess the effectiveness of a system across multiple queries, a commonly used measure is mean average precision (MAP). Mean average precision is defined as the arithmetic mean of per-topic average precision,  $MAP = \frac{1}{n} \sum_n AP_n$ . A consequence of the arithmetic mean is that, if a system improvement doubles AP for one topic from 0.02 to 0.04, while simultaneously decreasing AP on another from 0.4 to 0.38, the MAP will be unchanged. If we prefer to highlight performance differences on the lowest performing topics, a widely used alternative is the geometric mean of average precision (GMAP), first introduced in the TREC 2004 robust track (Voorhees, 2006).

$$GMAP = \sqrt[n]{\prod_n AP_n}$$

Robertson (2006) presents a justification and analysis of GMAP and notes that it may alternatively be computed as an arithmetic mean of logs,

$$GMAP = \exp \frac{1}{n} \sum_n \log AP_n.$$

#### 4.2.2 Significance Testing for GMAP

A standard way of measuring the significance of system improvements in MAP is to compare average precision (AP) on each of the evaluation queries

using the Wilcoxon signed-rank test. This test, while not requiring a particular distribution on the measurements, does assume that they belong to an interval scale. Similarly, the arithmetic mean of MAP assumes AP has interval scale. As Robertson (2006) has pointed out, it is in no sense clear that AP (prior to any transformation) satisfies this assumption. This becomes an argument for GMAP, since it may also be defined using an arithmetic mean of log-transformed average precisions. That is to say, the logarithm is simply one possible monotonic transformation which is arguably as good as any other, including the identify transform, in terms of whether the transformed value satisfies the interval assumption. This log transform (and hence GMAP) is useful simply because it highlights improvements on the most difficult queries.

We apply the same reasoning to test for statistical significance in GMAP improvements. That is, we test for significant improvements in GMAP by applying the Wilcoxon signed rank test to the paired, transformed average precisions, log AP. We handle tied pairs and compute exact  $p$ -values using the Streitberg & Röhmel Shift-Algorithm (1990). For topics with  $AP = 0$ , we follow the Robust Track convention and add  $\epsilon = 0.00001$ . The authors are not aware of significance tests having been previously reported on GMAP.

### 4.3 Retrieval System

We use Okapi BM25 (Robertson et al., 1996) as our basic retrieval system, which defines a document  $D$ ’s retrieval score for query  $Q$  as

$$s(D, Q) = \sum_{i=1}^n idf(q_i) \frac{\left(\frac{k_3+1}{k_3+qf_i}\right) f(q_i, D) (k_1 + 1)}{f(q_i, D) + k_1(1 - b + b \frac{|D|}{avgdl})},$$

where the inverse document frequency ( $idf$ ) is defined as

$$idf(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5},$$

$N$  is the size of the collection,  $n(q_i)$  is the document frequency for term  $q_i$ ,  $qf_i$  is the frequency of term  $q_i$  in query  $Q$ ,  $f(q_i, D)$  is the term frequency of query term  $q_i$  in document  $D$ ,  $|D|$  is the length of the matching document, and  $avgdl$  is the average length of a document in the collection. We set the

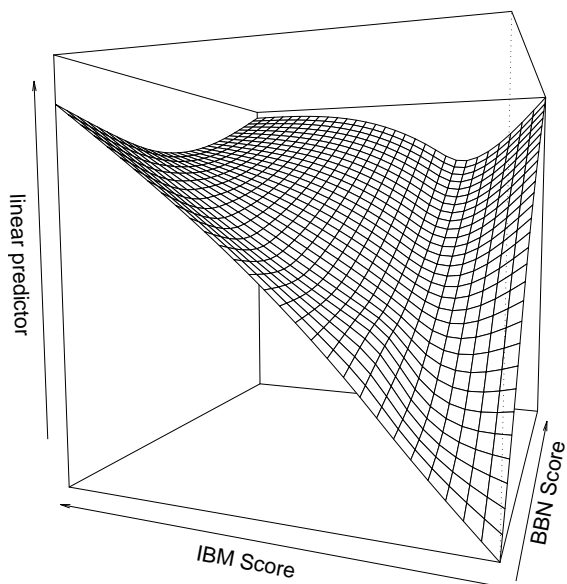


Figure 2: The two dimensional smooth  $f(s_{IBM}, s_{BBN})$  learned to predict relevance given input scores from IBM and BBN transcripts.

parameters to  $k_1 = 1$ ,  $k_3 = 1$ ,  $b = .5$ , which gave good results on a single transcript.

#### 4.4 Speech Recognition Transcripts

Our first set of speech recognition transcripts was produced by IBM for the MALACH project, and used for several years in the CLEF cross-language speech retrieval (CL-SR) track (Pecina et al., 2007). The IBM recognizer was built using a manually produced pronunciation dictionary and 200 hours of transcribed audio. The resulting interview transcripts have a reported mean word error rate (WER) of approximately 25% on held out data, which was obtained by priming the language model with metadata available from pre-interview questionnaires. This represents significant improvements over IBM transcripts used in earlier CL-SR evaluations, which had a best reported WER of 39.6% (Byrne et al., 2004). This system is reported to have run at approximately 10 times real time.

##### 4.4.1 New Transcripts for MALACH

We were graciously permitted to use BBN Technology’s speech recognition system to produce a second set of ASR transcripts for our experiments (Prasad et al., 2005; Matsoukas et al., 2005). We selected the one side of the audio having largest RMS

amplitude for training and decoding. This channel was down-sampled to 8kHz and segmented using an available broadcast news segmenter. Because we did not have a pronunciation dictionary which covered the transcribed audio, we automatically generated pronunciations for roughly 14k words using a rule-based transliterator and the CMU lexicon. Using the same 200 hours of transcribed audio, we trained acoustic models as described in (Prasad et al., 2005). We use a mixture of the training transcripts and various newswire sources for our language model training. We did not attempt to prime the language model for particular interviewees or otherwise utilize any interview metadata. For decoding, we ran a fast (approximately 1 times real time) system, as described in (Matsoukas et al., 2005). Unfortunately, as we do not have the same development set used by IBM, a direct comparison of WER is not possible. Testing on a small held out set of 4.3 hours, we observed our system had a WER of 32.4%.

#### 4.5 Combination Methods

For baseline comparisons, we ran our evaluation on each of the two transcript sources (IBM and our new transcripts), the linear combination chosen to optimize MAP (LC-MAP), the linear combination chosen to optimize GMAP (LC-GMAP), interleaving (IL), and CombMNZ. We denote our additive factor model as Factor GLM, and our multidimensional smooth GAM model as MD-GAM.

Linear combination parameters were chosen to optimize performance on the training set, sweeping the weight for each source at intervals of 0.01. For the generalized additive models, we maximized the penalized likelihood of the training examples under our model, as described in Section 3.

## 5 Results

Table 1 shows our complete set of results. This includes baseline scores from our new set of transcripts, each of our baseline combination approaches, and results from our proposed combination models. Although we are chiefly interested in improvements on difficult topics (i.e., GMAP), we present MAP for comparison. Results in bold indicate the largest mean value of the measure (either AP or log AP), while daggers (†) indicate the

Type	Model	MAP	GMAP
T	IBM	0.0531 (-.2)	0.0134 (-11.8)
-	BBN	0.0532	0.0152
-	LC-MAP	0.0564 (+6.0)	0.0158 (+3.9)
-	LC-GMAP	0.0587 (+10.3)	0.0154 (+1.3)
-	IL	0.0592 (+11.3)	0.0165 (+8.6)
-	CombMNZ	0.0550 (+3.4)	0.0150 (-1.3)
-	Factor GLM	<b>0.0611 (+14.9)</b> <sup>†</sup>	0.0161 (+5.9)
-	MD-GAM	0.0561 (+5.5) <sup>†</sup>	<b>0.0180 (+18.4)</b> <sup>†</sup>
TD	IBM	0.0415 (-15.1)	0.0173 (-9.9)
-	BBN	0.0489	0.0192
-	LC-MAP	0.0519 (+6.1) <sup>†</sup>	0.0201 (+4.7) <sup>†</sup>
-	LC-GMAP	<b>0.0531 (+8.6)</b> <sup>†</sup>	0.0200 (+4.2)
-	IL	0.0507 (+3.7)	0.0210 (+9.4)
-	CombMNZ	0.0495 (+1.2) <sup>†</sup>	0.0196 (+2.1)
-	Factor GLM	0.0526 (+7.6) <sup>†</sup>	0.0198 (+3.1)
-	MD-GAM	0.0529 (+8.2) <sup>†</sup>	<b>0.0223 (+16.2)</b> <sup>†</sup>

Table 1: MAP and GMAP for each combination approach, using the evaluation query set from the CLEF-2007 CL-SR (MALACH) collection. Shown in parentheses is the relative improvement in score over the best single transcripts results (i.e., using our new set of transcripts). The best (mean) score for each condition is in bold.

combination is a statistically significant improvement ( $\alpha = 0.05$ ) over our new transcript set (that is, over the best single transcript result). Tests for statistically significant improvements in GMAP are computed using our paired log AP test, as discussed in Section 4.2.2.

First, we note that the GAM model with multi-dimensional smooth gives the largest GMAP improvement for both title and title-description runs. Secondly, it is the only combination approach able to produce statistically significant relative improvements on both measures for both conditions. For GMAP, our measure of interest, these improvements are 18.4% and 16.2% respectively.

One surprising observation from Table 1 is that the mean improvement in log AP for interleaving is fairly large and yet not statistically significant (it is in fact a larger *mean* improvement than several other baseline combination approaches which *are* significant improvements. This may suggest that interleaving suffers from a large disparity between its best and worst performance on the query set.

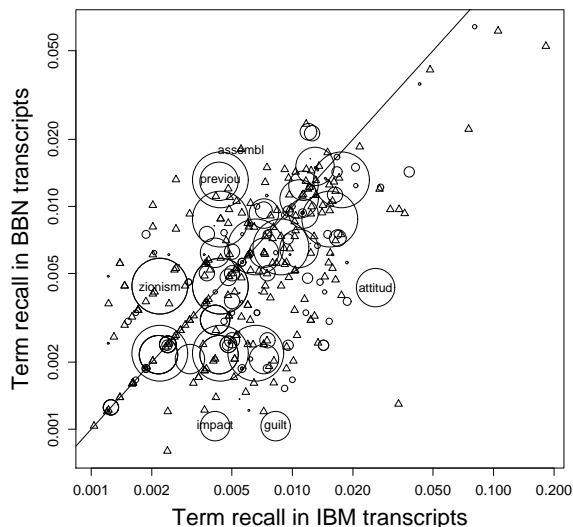


Figure 3: The proportion of relevant documents returned in IBM and BBN transcripts for discriminative title words (title words occurring in less than .01 of the collection). Point size is proportional to the improvement in average precision using (1) the best linear combination chosen to optimize GMAP ( $\triangle$ ) and (2) the combination using MD-GAM ( $\circ$ ).

Figure 3 examines whether our improvements come systematically from only one of the transcript sources. It shows the proportion of relevant documents in each transcript source containing the most discriminative title words (words occurring in less than .01 of the collection). Each point represents one term for one topic. The size of the point is proportional to the difference in AP observed on that topic by using MD-GAM and by using LC-GMAP. If the difference is positive (MD-GAM wins), we plot  $\circ$ , otherwise  $\triangle$ . First, we observe that, when it wins, MD-GAM tends to increase AP much more than when LC-GMAP wins. While there are many wins also for LC-GMAP, the effects of the larger MD-GAM improvements will dominate for many of the most difficult queries. Secondly, there does not appear to be any evidence that one transcript source has much higher term-recall than the other.

## 5.1 Oracle linear combination

A chief advantage of our MD-GAM combination model is that it is able to map input scores nonlinearly onto a probability of document relevance.

Type	Model	GMAP
T	Oracle-LC-GMAP	0.0168
-	MD-GAM	<b>0.0180 (+7.1)</b>
TD	Oracle-LC-GMAP	0.0222
-	MD-GAM	<b>0.0223 (+0.5)</b>

Table 2: GMAP results for an oracle experiment in which MD-GAM was fairly trained and LC-GMAP was unfairly optimized on the test queries.

To make an assessment of how much this capability helps the system, we performed an oracle experiment where we again constrained MD-GAM to be fairly trained but allowed LC-GMAP to cheat and choose the combination *optimizing* GMAP *on the test data*. Table 2 lists the results. While the improvement with MD-GAM is now not statistically significant (primarily because of our small query set), we found it still out-performed the oracle linear combination. For title-only queries, this improvement was surprisingly large at 7.1% relative.

## 6 Conclusion

While speech retrieval is one example of retrieval under errorful document representations, other similar tasks may also benefit from these combination models. This includes the task of cross-language retrieval, as well as the retrieval of documents obtained by optical character recognition.

Within speech retrieval, further work also remains to be done. For example, various other features are likely to be useful in predicting optimal system combination. These might include, for example, confidence scores, acoustic confusability, or other strong cues that one recognition system is unlikely to have properly recognized a query term. We look forward to investigating these possibilities in future work.

The question of how much a system should expose its internal workings (e.g., its document representations) to external systems is a long standing problem in meta-search. We've taken the rather narrow view that systems might only expose the list of scores they assigned to retrieved documents, a plausible scenario considering the many systems now emerging which are effectively doing this already. Some examples include *EveryZing*,<sup>1</sup> the MIT *Lec-*

<sup>1</sup><http://www.everyzing.com/>

*ture Browser*,<sup>2</sup> and Comcast's video search.<sup>3</sup> This trend is likely to continue as the underlying representations of the content are themselves becoming increasingly complex (e.g., word and subword level lattices or confusion networks). The cost of exposing such a vast quantity of such complex data rapidly becomes difficult to justify.

But if the various representations of the content are available, there are almost certainly other combination approaches worth investigating. Some possible approaches include simple linear combinations of the putative term frequencies, combinations of one best transcript hypotheses (e.g., using ROVER (Fiscus, 1997)), or methods exploiting word-lattice information (Evermann and Woodland, 2000).

Our planet's 6.6 billion people speak many more words every day than even the largest Web search engines presently index. While much of this is surely not worth hearing again (or even once!), some of it is surely precious beyond measure. Separating the wheat from the chaff in this cacophony is the *raison d'etre* for information retrieval, and it is hard to conceive of an information retrieval challenge with greater scope or greater potential to impact our society than improving our access to the spoken word.

## Acknowledgements

The authors are grateful to BBN Technologies, who generously provided access to their speech recognition system for this research.

## References

- Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. 1994. Automatic combination of multiple ranked retrieval systems. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 173–181.
- Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, David Grossman, Ophir Frieder, and Nazli Goharian. 2004. Fusion of effective retrieval strategies in the same information retrieval system. *J. Am. Soc. Inf. Sci. Technol.*, 55(10):859–868.
- W. Byrne, D. Doermann, M. Franz, S. Gustman, J. Hajic, D.W. Oard, M. Picheny, J. Psutka, B. Ramabhadran,

<sup>2</sup><http://web.sls.csail.mit.edu/lectures/>

<sup>3</sup><http://videosearch.comcast.net>



- D. Soergel, T. Ward, and Wei-Jing Zhu. 2004. Automatic recognition of spontaneous speech for access to multilingual oral history archives. *IEEE Transactions on Speech and Audio Processing, Special Issue on Spontaneous Speech Processing*, 12(4):420–435, July.
- J. P. Callan, Z. Lu, and W. Bruce Croft. 1995. Searching Distributed Collections with Inference Networks. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington. ACM Press.
- G. Evermann and P.C. Woodland. 2000. Posterior probability decoding, confidence estimation and system combination. In *Proceedings of the Speech Transcription Workshop*, May.
- Jonathan G. Fiscus. 1997. A Post-Processing System to Yield Reduced Word Error Rates: Recogniser Output Voting Error Reduction (ROVER). In *Proceedings of the IEEE ASRU Workshop*, pages 347–352.
- Jong-Hak Lee. 1997. Analyses of multiple evidence combination. In *SIGIR Forum*, pages 267–276.
- David Lillis, Fergus Toolan, Rem Collier, and John Dunning. 2006. Probuse: a probabilistic approach to data fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 139–146, New York, NY, USA. ACM.
- R. Manmatha, T. Rath, and F. Feng. 2001. Modeling score distributions for combining the outputs of search engines. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267–275, New York, NY, USA. ACM.
- Spyros Matsoukas, Rohit Prasad, Srinivas Laxminarayan, Bing Xiang, Long Nguyen, and Richard Schwartz. 2005. The 2004 BBN 1xRT Recognition Systems for English Broadcast News and Conversational Telephone Speech. In *Interspeech 2005*, pages 1641–1644.
- Mark Montague and Javed A. Aslam. 2002. Condorcet fusion for improved retrieval. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 538–548, New York, NY, USA. ACM.
- Pavel Pecina, Petra Hoffmannova, Gareth J.F. Jones, Jianqiang Wang, and Douglas W. Oard. 2007. Overview of the CLEF-2007 Cross-Language Speech Retrieval Track. In *Proceedings of the CLEF 2007 Workshop on Cross-Language Information Retrieval and Evaluation*, September.
- Allison L. Powell, James C. French, James P. Callan, Margaret E. Connell, and Charles L. Viles. 2000. The impact of database selection on distributed searching. In *Research and Development in Information Retrieval*, pages 232–239.
- R. Prasad, S. Matsoukas, C.L. Kao, J. Ma, D.X. Xu, T. Colthurst, O. Kimball, R. Schwartz, J.L. Gauvain, L. Lamel, H. Schwenk, G. Adda, and F. Lefevre. 2005. The 2004 BBN/LIMSI 20xRT English Conversational Telephone Speech Recognition System. In *Interspeech 2005*.
- S. Robertson, S. Walker, S. Jones, and M. Hancock-Beaulieu M. Gatford. 1996. Okapi at TREC-3. In *Text REtrieval Conference*, pages 21–30.
- Stephen Robertson. 2006. On GMAP: and other transformations. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 78–83, New York, NY, USA. ACM.
- J. Savoy, A. Le Calvé, and D. Vrajitoru. 1988. Report on the TREC-5 experiment: Data fusion and collection fusion.
- Joseph A. Shaw and Edward A. Fox. 1994. Combination of multiple searches. In *Proceedings of the 2nd Text REtrieval Conference (TREC-2)*.
- Bernd Streitberg and Joachim Röhmel. 1990. On tests that are uniformly more powerful than the Wilcoxon-Mann-Whitney test. *Biometrics*, 46(2):481–484.
- Christopher C. Vogt and Garrison W. Cottrell. 1999. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173.
- Ellen M. Voorhees, Narendra Kumar Gupta, and Ben Johnson-Laird. 1994. The collection fusion problem. In D. K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, pages 500–225. National Institute of Standards and Technology.
- Ellen M. Voorhees. 2006. Overview of the TREC 2005 robust retrieval track. In Ellen M. Voorhees and L.P. Buckland, editors, *The Fourteenth Text REtrieval Conference, (TREC 2005)*, Gaithersburg, MD: NIST.
- Simon N. Wood. 2003. Thin plate regression splines. *Journal Of The Royal Statistical Society Series B*, 65(1):95–114.
- Simon Wood. 2006. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC.

# A Critical Reassessment of Evaluation Baselines for Speech Summarization

Gerald Penn and Xiaodan Zhu

University of Toronto  
10 King's College Rd.  
Toronto M5S 3G4 CANADA  
{gpenn, xzhu}@cs.toronto.edu

## Abstract

We assess the current state of the art in speech summarization, by comparing a typical summarizer on two different domains: lecture data and the SWITCHBOARD corpus. Our results cast significant doubt on the merits of this area's accepted evaluation standards in terms of: baselines chosen, the correspondence of results to our intuition of what "summaries" should be, and the value of adding speech-related features to summarizers that already use transcripts from automatic speech recognition (ASR) systems.

## 1 Problem definition and related literature

Speech is arguably the most basic, most natural form of human communication. The consistent demand for and increasing availability of spoken audio content on web pages and other digital media should therefore come as no surprise. Along with this availability comes a demand for ways to better navigate through speech, which is inherently more linear or sequential than text in its traditional delivery.

*Navigation* connotes a number of specific tasks, including search, but also browsing (Hirschberg et al., 1999) and skimming, which can involve far more analysis and manipulation of content than the spoken document retrieval tasks of recent NIST fame (1997 2000). These would include time compression of the speech signal and/or "dichotic" presentations of speech, in which a different audio track is presented to either ear (Cherry and Taylor, 1954; Ranjan et al., 2006). Time compression of speech, on the other hand, excises small slices of digitized

speech data out of the signal so that the voices speak all of the content but more quickly. The excision can either be fixed rate, for which there have been a number of experiments to detect comprehension limits, or variable rate, where the rate is determined by pause detection and shortening (Arons, 1992), pitch (Arons, 1994) or longer-term measures of linguistic salience (Tucker and Whittaker, 2006). A very short-term measure based on spectral entropy can also be used (Ajmal et al., 2007), which has the advantage that listeners cannot detect the variation in rate, but they nevertheless comprehend better than fixed-rate baselines that preserve pitch periods. With or without variable rates, listeners can easily withstand a factor of two speed-up, but Likert response tests definitively show that they absolutely hate doing it (Tucker and Whittaker, 2006) relative to word-level or utterance-level excisive methods, which would include the summarization-based strategy that we pursue in this paper.

The strategy we focus on here is summarization, in its more familiar construal from computational linguistics and information retrieval. We view it as an extension of the text summarization problem in which we use automatically prepared, imperfect textual transcripts to summarize speech. Other details are provided in Section 2.2. Early work on speech summarization was either domain-restricted (Kameyama and Arima, 1994), or prided itself on not using ASR at all, because of its unreliability in open domains (Chen and Withgott, 1992). Summaries of speech, however, can still be delivered audially (Kikuchi et al., 2003), even when (noisy) transcripts are used.

The purpose of this paper is not so much to introduce a new way of summarizing speech, as to critically reappraise how well the current state of the art really works. The earliest work to consider open-domain speech summarization seriously from the standpoint of text summarization technology (Valenza et al., 1999; Zechner and Waibel, 2000) approached the task as one of speech transcription followed by text summarization of the resulting transcript (weighted by confidence scores from the ASR system), with the very interesting result that transcription and summarization errors in such systems tend to offset one another in overall performance. In the years following this work, however, some research by others on speech summarization (Maskey and Hirschberg, 2005; Murray et al., 2005; Murray et al., 2006, *inter alia*) has focussed *de rigueur* on striving for and measuring the improvements attainable over the transcribe-then-summarize baseline with features available from non-transcriptional sources (e.g., pitch and energy of the acoustic signal) or those, while evident in textual transcripts, not germane to texts other than spoken language transcripts (e.g., speaker changes or question-answer pair boundaries).

These “novel” features do indeed seem to help, but not by nearly as much as some of this recent literature would suggest. The experiments and the choice of baselines have largely been framed to illuminate the value of various knowledge sources (“prosodic features,” “named entity features” etc.), rather than to optimize performance *per se* — although the large-dimensional pattern recognition algorithms and classifiers that they use are inappropriate for descriptive hypothesis testing.

First, most of the benefit attained by these novel sources can be captured simply by measuring the lengths of candidate utterances. Only one paper we are aware of (Christensen et al., 2004) has presented the performance of length on its own, although the objective there was to use length, position and other simple textual feature baselines (no acoustics) to distinguish the properties of various genres of spoken audio content, a topic that we will return to in Section 2.1.<sup>1</sup> Second, maximal marginal relevance

(MMR) has also fallen by the wayside, although it too performs very well. Again, only one paper that we are aware of (Murray et al., 2005) provides an MMR baseline, and there MMR significantly outperforms an approach trained on a richer collection of features, including acoustic features. MMR was the method of choice for utterance selection in Zechner and Waibel (2000) and their later work, but it is often eschewed perhaps because textbook MMR does not directly provide a means to incorporate other features. There is a simple means of doing so (Section 2.3), and it is furthermore very resilient to low word-error rates (WERs, Section 3.3).

Third, as inappropriate uses of optimization methods go, the one comparison that has not made it into print yet is that of the more traditional “what-is-said” features (MMR, length in words and named-entity features) vs. the avant-garde “how-it-is-said” features (structural, acoustic/prosodic and spoken-language features). Maskey & Hirschberg (2005) divide their features into these categories, but only to compute a correlation coefficient between them (0.74). The former in aggregate still performs significantly better than the latter in aggregate, even if certain members of the latter do outperform certain members of the former. This is perhaps the most reassuring comparison we can offer to text summarization and ASR enthusiasts, because it corroborates the important role that ASR still plays in speech summarization in spite of its imperfections.

Finally, and perhaps most disconcertingly, we can show that current speech summarization performs just as well, and in some respects even better, with SWITCHBOARD dialogues as it does with more coherent spoken-language content, such as lectures. This is not a failing of automated systems themselves — even *humans* exhibit the same tendency under the experimental conditions that most researchers have used to prepare evaluation gold standards. What this means is that, while speech summarization systems may arguably be useful and are indeed consistent with whatever it is that humans are doing when they are enlisted to rank utterances, this evaluation regime simply does not reflect how well the “summaries” capture the goal-orientation or

---

<sup>1</sup>Length features are often mentioned in the text of other work as the most beneficial single features in more hetero-

---

geneous systems, but without indicating their performance on their own.

higher-level purpose of the data that they are trained on. As a community, we have been optimizing an utterance excerpting task, we have been moderately successful at it, but this task in at least one important respect bears no resemblance to what we could convincingly call speech summarization.

These four results provide us with valuable insight into the current state of the art in speech summarization: it is not summarization, the aspiration to measure the relative merits of knowledge sources has masked the prominence of some very simple baselines, and the Zechner & Waibel pipe-ASR-output-into-text-summarizer model is still very competitive — what seems to matter more than having access to the raw spoken data is simply knowing that it is spoken data, so that the most relevant, still textually available features can be used. Section 2 describes the background and further details of the experiments that we conducted to arrive at these conclusions. Section 3 presents the results that we obtained. Section 4 concludes by outlining an ecologically valid alternative for evaluating real summarization in light of these results.

## 2 Setting of the experiment

### 2.1 Provenance of the data

Speech summarizers are generally trained to summarize either broadcast news or meetings. With the exception of one paper that aspires to compare the “styles” of spoken and written language *ceteris paribus* (Christensen et al., 2004), the choice of broadcast news as a source of data in more recent work is rather curious. Broadcast news, while open in principle in its range of topics, typically has a range of closely parallel, written sources on those same topics, which can either be substituted for spoken source material outright, or at the very least be used corroboratively alongside them. Broadcast news is also read by professional news readers, using high quality microphones and studio equipment, and as a result has very lower WER — some even call ASR a solved problem on this data source. Broadcast news is also very text-like at a deeper level. Relative position within a news story or dialogue, the dreaded baseline of text summarization, works extremely well in spoken broadcast news summarization, too. Within the operating region of the receiver

operating characteristics (ROC) curve most relevant to summarizers (0.1–0.3), Christensen et al. (2004) showed that position was by far the best feature in a read broadcast news system with high WER, and that position and length of the extracted utterance were the two best with low WER. Christensen et al. (2004) also distinguished read news from “spontaneous news,” broadcasts that contain interviews and/or man-in-the-field reports, and showed that in the latter variety position is not at all prominent at any level of WER, but length is. Maskey & Hirschberg’s (2005) broadcast news is a combination of read news and spontaneous news.

Spontaneous speech, in our view, particularly in the lecture domain, is our best representative of what needs to be summarized. Here, the positional baseline performs quite poorly (although length does extremely well, as discussed below), and ASR performance is far from perfect. In the case of lectures, there are rarely exact transcripts available, but there are bulleted lines from presentation slides, related research papers on the speaker’s web page and monographs on the same topic that can be used to improve the language models for speech recognition systems. Lectures have just the right amount of props for realistic ASR, but still very open domain vocabularies and enough spontaneity to make this a problem worth solving. As discussed further in Section 4, the classroom lecture genre also provides us with a task that we hope to use to conduct a better grounded evaluation of real summarization quality.

To this end, we use a corpus of lectures recorded at the University of Toronto to train and test our summarizer. Only the lecturer is recorded, using a head-worn microphone, and each lecture lasts 50 minutes. The lectures in our experiments are all undergraduate computer science lectures. The results reported in this paper used four different lectures, each from a different course and spoken by a different lecturer. We used a leave-one-out cross-validation approach by iteratively training on three lectures worth of material and testing on the one remaining. We combine these iterations by averaging. The lectures were divided at random into 8–15 minute intervals, however, in order to provide a better comparison with the SWITCHBOARD dialogues. Each interval was treated as a separate document and was summarized separately. So the four lectures together actually

provide 16 SWITCHBOARD-sized samples of material, and our cross-validation leaves on average four of them out in a turn.

We also use part of the SWITCHBOARD corpus in one of our comparisons. SWITCHBOARD is a collection of telephone conversations, in which two participants have been told to speak on a certain topic, but with no objective or constructive goal to proceed towards. While the conversations are locally coherent, this lack of goal-orientation is acutely apparent in all of them — they may be as close as any speech recording can come to being about nothing.<sup>2</sup> We randomly selected 27 conversations, containing a total of 3665 utterances (identified by pause length), and had three human annotators manually label each utterance as in- or out-of-summary. Interestingly, the interannotator agreement on SWITCHBOARD ( $\kappa = 0.383$ ) is higher than on the lecture corpus (0.372) and higher than the  $\kappa$ -score reported by Galley (2006) for the ICSI meeting data used by Murray et al. (2005; 2006), in spite of the fact that Murray et al. (2005) primed their annotators with a set of questions to consider when annotating the data.<sup>3</sup> This does not mean that the SWITCHBOARD summaries are qualitatively better, but rather that annotators are apt to agree more on which utterances to include in them.

## 2.2 Summarization task

As with most work in speech summarization, our strategy involves considering the problem as one of utterance extraction, which means that we are not synthesizing new text or speech to include in summaries, nor are we attempting to extract small phrases to sew together with new prosodic contours. Candidate utterances are identified through pause-length detection, and the length of these pauses has been experimentally calibrated to 200 msec, which results in roughly sentence-sized utterances. Summarization then consists of choosing the best N% of these utterances for the summary, where N is typ-

<sup>2</sup>It should be noted that the meandering style of SWITCHBOARD conversations does have correlates in text processing, particularly in the genres of web blogs and newsgroup- or wiki-based technical discussions.

<sup>3</sup>Although we did define what a summary was to each annotator beforehand, we did not provide questions or suggestions on content for either corpus.

ically between 10 and 30. We will provide ROC curves to indicate performance as a function over all N. An ROC is plotted along an x-axis of *specificity* (true-negative-rate) and a y-axis of *sensitivity* (true-positive-rate). A larger area under the ROC corresponds to better performance.

## 2.3 Utterance isolation

The framework for our extractive summarization experiments is depicted in Figure 1. With the exception of disfluency removal, it is very similar in its overall structure to that of Zechner’s (2001). The summarizer takes as input either manual or automatic transcripts together with an audio file, and has three modules to process disfluencies and extract features important to identifying sentences.

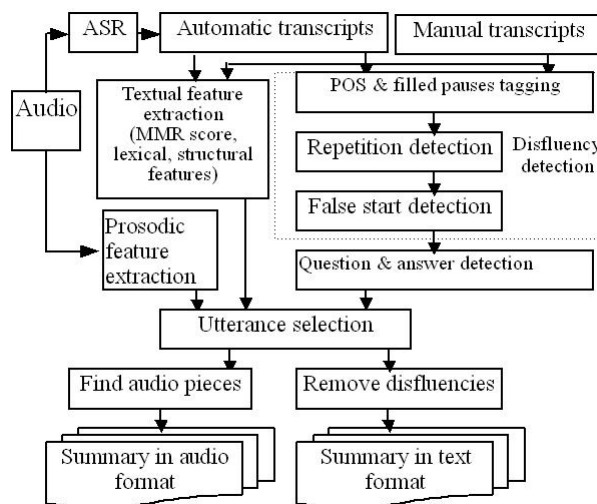


Figure 1: Experimental framework for summarizing spontaneous conversations.

During sentence boundary detection, words that are likely to be adjacent to an utterance boundary are determined. We call these words *trigger words*.

False starts are very common in spontaneous speech. According to Zechner’s (2001) statistics on the SWITCHBOARD corpus, they occur in 10-15% of all utterances. A decision tree (C4.5, Release 8) is used to detect false starts, trained on the POS tags and trigger-word status of the first and last four words of sentences from a training set. Once false starts are detected, these are removed.

We also identify repetitions as a sequence of between 1 and 4 words which is consecutively re-

peated in spontaneous speech. Generally, repetitions are discarded. Repetitions of greater length are extremely rare statistically and are therefore ignored.

Question-answer pairs are also detected and linked. Question-answer detection is a two-stage process. The system first identifies the questions and then finds the corresponding answer. For (both WH- and Yes/No) question identification, another C4.5 classifier was trained on 2,000 manually annotated sentences using utterance length, POS bigram occurrences, and the POS tags and trigger-word status of the first and last five words of an utterance. After a question is identified, the immediately following sentence is labelled as the answer.

## 2.4 Utterance selection

To obtain a trainable utterance selection module that can utilize and compare rich features, we formulated utterance selection as a standard binary classification problem, and experimented with several state-of-the-art classifiers, including linear discriminant analysis LDA, support vector machines with a radial basis kernel (SVM), and logistic regression (LR), as shown in Figure 2 (computed on SWITCHBOARD data). MMR, Zechner’s (2001) choice, is provided as a baseline. MMR linearly interpolates a relevance component and a redundancy component that balances the need for new vs. salient information. These two components can just as well be mixed through LR, which admits the possibility of adding more features and the benefit of using LR over held-out estimation.

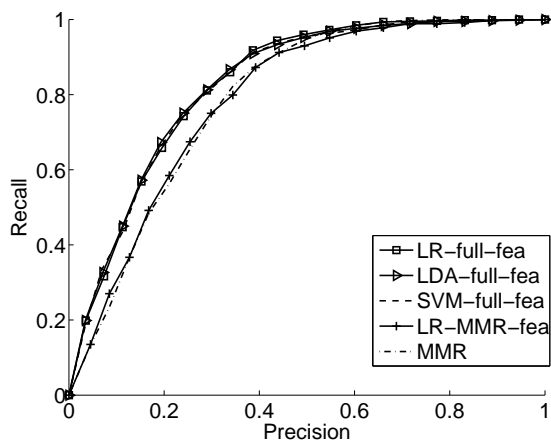


Figure 2: Precision-recall curve for several classifiers on the utterance selection task.

As Figure 2 indicates, there is essentially no difference in performance among the three classifiers we tried, nor between MMR and LR restricted to the two MMR components. This is important, since we will be comparing MMR to LR-trained classifiers based on other combinations of features below. The ROC curves in the remainder of this paper have been prepared using the LR classifier.

## 2.5 Features extracted

While there is very little difference realized across pattern recognition methods, there is much more at stake with respect to which features the methods use to characterize their input. We can extract and use the features in Figure 3, arranged there according to their knowledge source.

We detect disfluencies in the same manner as Zechner (2001)). Taking ASR transcripts as input, we use the Brill tagger (Brill, 1995) to assign POS tags to each word. There are 42 tags: Brill’s 38 plus four which identify filled-pause disfluencies:

- empty coordinating conjunctions (CO),
- lexicalized filled pauses (DM),
- editing terms (ET), and
- non-lexicalized filled pauses (UH).

Our disfluency features include the number of each of these, their total, and also the number of repetitions. Disfluencies adjacent to a speaker turn are ignored, however, because they occur as a normal part of turn coordination between speakers.

Our preliminary experiments suggest that speaker meta-data do not improve on the quality of summarization, and so this feature is not included.

We indicate with bold type the features that indicate some quantity of length, and we will consider these as members of another class called “length,” in addition to their given class above. In all of the data on which we have measured, the correlation between time duration and number of words is nearly 1.00 (although pause length is not).

## 2.6 Evaluation of summary quality

We plot receiver operating characteristic (ROC) curves along a range of possible compression parameters, and in one case, ROUGE scores. ROUGE

1. Lexical features
    - MMR score<sup>4</sup>,
    - **utterance length (in words)**,
  2. Named entity features — number of:
    - person names,
    - location names
    - organization names
    - the sum of these
  3. Structural features
    - utterance position, labelled as first, middle, or last one-third of the conversation
    - a Boolean feature indicating whether an utterance is adjacent to a speaker turn
1. Acoustic features — min, max and avg. of:<sup>5</sup>
    - pitch
    - energy
    - speaking rate
    - **(unfilled) pause length**
    - **time duration (in msec)**
  2. “Spoken language” features
    - disfluencies
    - given/new information
    - question/answer pair identification

Figure 3: Features available for utterance selection by knowledge source. Features in bold type quantify length. In our experiments, we exclude these from their knowledge sources, and study them as a separate length category.

and F-measure are both widely used in speech summarization, and they have been shown by others to be broadly consistent on speech summarization tasks (Zhu and Penn, 2005).

### 3 Results and analysis

#### 3.1 Lecture corpus

The results of our evaluation on the lecture data appear in Figure 4. As is evident, there is very little difference among the combinations of features with this data source, apart from the positional baseline, “lead,” which simply chooses the first N% of the utterances. This performs quite poorly. The best performance is achieved by using all of the features together, but the length baseline, which uses only those features in bold type from Figure 3, is very close (no statistically significant difference), as is MMR.<sup>6</sup>

<sup>4</sup>When evaluated on its own, the MMR interpolating parameter is set through experimentation on a held-out dataset, as in Zechner (2001). When combined with other features, its relevance and redundancy components are provided to the classifier separately.

<sup>5</sup>All of these features are calculated on the word level and normalized by speaker.

<sup>6</sup>We conducted the same evaluation without splitting the lectures into 8–15 minute segments (so that the summaries summarize an entire lecture), and although space here precludes the presentation of the ROC curves, they are nearly identical

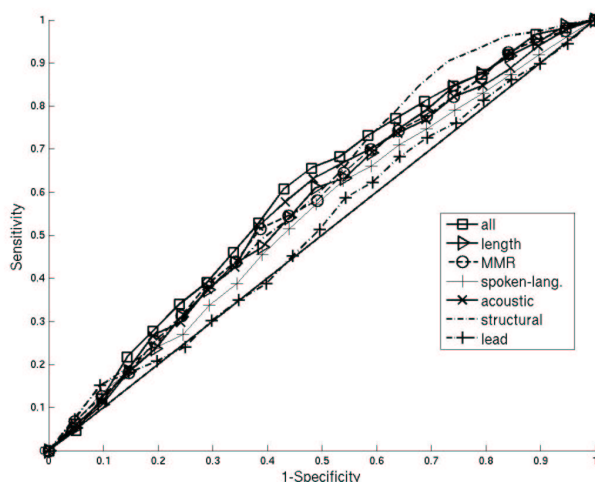


Figure 4: ROC curve for utterance selection with the lecture corpus with several feature combinations.

#### 3.2 SWITCHBOARD corpus

The corresponding results on SWITCHBOARD are shown in Figure 5. Again, length and MMR are very close to the best alternative, which is again all of features combined. The difference with respect to either of these baselines is statistically significant within the popular 10–30% compression range, as is the classifier trained on all features but acoustic to those on the segments shown here.

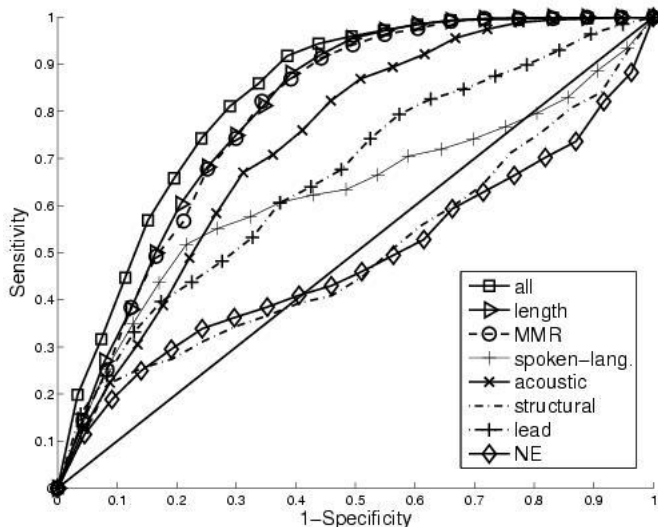


Figure 5: ROC curve for SWITCHBOARD utterance selection with several feature combinations.

(not shown). The classifier trained on all features but spoken language features (not shown) is not significantly better, so it is the spoken language features that make the difference, not the acoustic features. The best score is also significantly better than on the lecture data, however, particularly in the 10–30% range. Our analysis of the difference suggests that the much greater variance in utterance length in SWITCHBOARD is what accounts for the overall better performance of the automated system as well as the higher human interannotator agreement. This also goes a long way to explaining why the length baseline is so good.

Still another perspective is to classify features as either “what-is-said” (MMR, length and NE features) or “how-it-is-said” (structural, acoustic and spoken-language features), as shown in Figure 6. What-is-said features are better, but only barely so within the usual operating region of summarizers.

### 3.3 Impact of WER

Word error rates (WERs) arising from speech recognition are usually much higher in spontaneous conversations than in read news. Having trained ASR models on SWITCHBOARD section 2 data with our sample of 27 conversations removed, the WER on that sample is 46%. We then train a language model on SWITCHBOARD section 2 without removing the 27-conversation sample so as to delib-

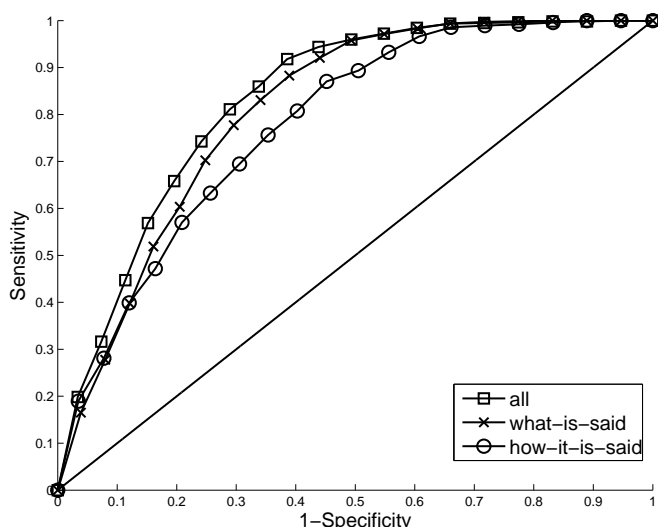


Figure 6: ROC curves for textual and non-textual features.

erately overfit the model. This pseudo-WER is then 39%. We might be able to get less WER by tuning the ASR models or by using more training data, but that is not the focus here. Summarizing the automatic transcripts generated from both of these systems using our LR-based classifier with all features, as well as manual (perfect) transcripts, we obtain the ROUGE-1 scores in Table 1.

WER	10%	15%	20%	25%	30%
0.46	.615	.591	.556	.519	.489
0.39	.615	.591	.557	.526	.491
0	.619	.600	.566	.530	.492

Table 1: ROUGE-1 of LR system with all features under different WERs.

Table 1 shows that WERs do not impact summarization performance significantly. One reason is that the acoustic and structural features are not affected by word errors, although WERs can affect the MMR, spoken language, length and NE features. Figures 7 and 8 present the ROC curves of the MMR and spoken language features, respectively, under different WERs. MMR is particularly resilient, even on SWITCHBOARD. Keywords are still often correctly recognized, even in the presence of high WER, although possibly because the same topic is discussed in many SWITCHBOARD conversations.



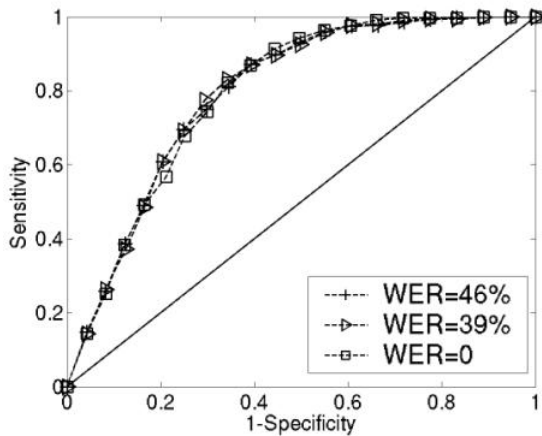


Figure 7: ROC curves for the effectiveness of MMR scores on transcripts under different WERs.

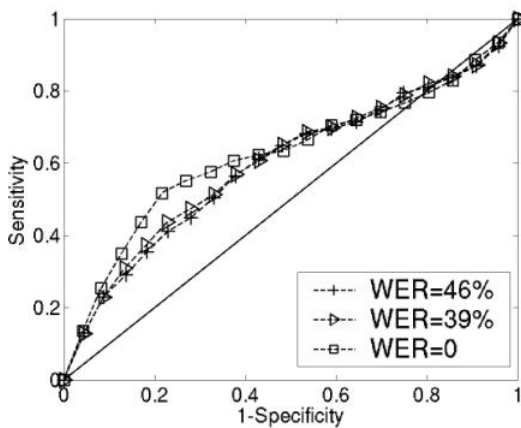


Figure 8: ROC curves for the effectiveness of spoken language features on transcripts under different WERs.

When some keywords are misrecognized (e.g. hat), furthermore, related words (e.g. dress, wear) still may identify important utterances. As a result, a high WER does not necessarily mean a worse transcript for bag-of-keywords applications like summarization and classification, regardless of the data source. Utterance length does not change very much when WERs vary, and in addition, it is often a latent variable that underlies some other features' role, e.g., a long utterance often has a higher MMR score than a short utterance, even when the WER changes.

Note that the effectiveness of spoken language features varies most between manually and automatically generated transcripts just at around the typi-

cal operating region of most summarization systems. The features of this category that respond most to WER are disfluencies. Disfluency detection is also at its most effective in this same range with respect to any transcription method.

#### 4 Future Work

In terms of future work in light of these results, clearly the most important challenge is to formulate an experimental alternative to measuring against a subjectively classified gold standard in which annotators are forced to commit to relative salience judgements with no attention to goal orientation and no requirement to synthesize the meanings of larger units of structure into a coherent message. It is here that using the lecture domain offers us some additional assistance. Once these data have been transcribed and outlined, we will be able to formulate examinations for students that test their knowledge of the topics being lectured upon: both their higher-level understanding of goals and conceptual themes, as well as factoid questions on particular details. A group of students can be provided with access to a collection of entire lectures to establish a theoretical limit. Experimental and control groups can then be provided with access only to summaries of those lectures, prepared using different sets of features, or different modes of delivery (text vs. speech), for example. This task-based protocol involves quite a bit more work, and at our university, at least, there are regulations that preclude us placing a group of students in a class at a disadvantage with respect to an examination for credit that need to be dealt with. It is, however, a far better means of assessing the quality of summaries in an ecologically valid context.

It is entirely possible that, within this protocol, the baselines that have performed so well in our experiments, such as length or, in read news, position, will utterly fail, and that less traditional acoustic or spoken language features will genuinely, and with statistical significance, add value to a purely transcript-based text summarization system. To date, however, that case has not been made. He et al. (1999) conducted a study very similar to the one suggested above and found no significant difference between using pitch and using slide transition boundaries. No ASR transcripts or length features were used.

## References

- M. Ajmal, A. Kushki, and K. N. Plataniotis. 2007. Time-compression of speech in informational talks using spectral entropy. In *Proceedings of the 8th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS-07)*.
- B. Arons. 1992. Techniques, perception, and applications of time-compressed speech. In *American Voice I/O Society Conference*, pages 169–177.
- B. Arons. 1994. *Speech Skimmer: Interactively Skimming Recorded Speech*. Ph.D. thesis, MIT Media Lab.
- E. Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.
- F. Chen and M. Withgott. 1992. The use of emphasis to automatically summarize a spoken discourse. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 229–232.
- E. Cherry and W. Taylor. 1954. Some further experiments on the recognition of speech, with one and two ears. *Journal of the Acoustic Society of America*, 26:554–559.
- H. Christensen, B. Kolluru, Y. Gotoh, and S. Renals. 2004. From text summarisation to style-specific summarisation for broadcast news. In *Proceedings of the 26th European Conference on Information Retrieval (ECIR-2004)*, pages 223–237.
- M. Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*.
- L. He, E. Sanocki, A. Gupta, and J. Grudin. 1999. Auto-summarization of audio-video presentations. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 489–498.
- J. Hirschberg, S. Whittaker, D. Hindle, F. Pereira, and A. Singhal. 1999. Finding information in audio: A new paradigm for audio browsing and retrieval. In *Proceedings of the ESCA/ETRW Workshop on Accessing Information in Spoken Audio*, pages 117–122.
- M. Kameyama and I. Arima. 1994. Coping with aboutness complexity in information extraction from spoken dialogues. In *Proceedings of the 3rd International Conference on Spoken Language Processing (ICSLP)*, pages 87–90.
- T. Kikuchi, S. Furui, and C. Hori. 2003. Two-stage automatic speech summarization by sentence extraction and compaction. In *Proceedings of the ISCA/IEEE Workshop on Spontaneous Speech Processing and Recognition (SSPR)*, pages 207–210.
- S. Maskey and J. Hirschberg. 2005. Comparing lexical, acoustic/prosodic, discourse and structural features for speech summarization. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Eurospeech)*, pages 621–624.
- G. Murray, S. Renals, and J. Carletta. 2005. Extractive summarization of meeting recordings. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Eurospeech)*, pages 593–596.
- G. Murray, S. Renals, J. Moore, and J. Carletta. 2006. Incorporating speaker and discourse features into speech summarization. In *Proceedings of the Human Language Technology Conference - Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 367–374.
- National Institute of Standards. 1997–2000. Proceedings of the Text REtrieval Conferences. <http://trec.nist.gov/pubs.html>.
- Abhishek Ranjan, Ravin Balakrishnan, and Mark Chignell. 2006. Searching in audio: the utility of transcripts, dichotic presentation, and time-compression. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 721–730, New York, NY, USA. ACM Press.
- S. Tucker and S. Whittaker. 2006. Time is of the essence: an evaluation of temporal compression algorithms. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 329–338, New York, NY, USA. ACM Press.
- R. Valenza, T. Robinson, M. Hickey, and R. Tucker. 1999. Summarization of spoken audio through information extraction. In *Proceedings of the ESCA/ETRW Workshop on Accessing Information in Spoken Audio*, pages 111–116.
- K. Zechner and A. Waibel. 2000. Minimizing word error rate in textual summaries of spoken language. In *Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP/NAACL)*, pages 186–193.
- K. Zechner. 2001. *Automatic Summarization of Spoken Dialogues in Unrestricted Domains*. Ph.D. thesis, Carnegie Mellon University.
- X. Zhu and G. Penn. 2005. Evaluation of sentence selection for speech summarization. In *Proceedings of the RANLP workshop on Crossing Barriers in Text Summarization Research*, pages 39–45.

# Intensional Summaries as Cooperative Responses in Dialogue: Automation and Evaluation

**Joseph Polifroni**

Nokia Research Center  
3 Cambridge Center  
Cambridge, MA 02142, USA  
joseph.polifroni@nokia.com

**Marilyn Walker**

Department of Computer Science  
University of Sheffield  
Sheffield, S1 4DP, United Kingdom  
M.A.Walker@sheffield.ac.uk

## Abstract

Despite its long history, and a great deal of research producing many useful algorithms and observations, research in cooperative response generation has had little impact on the recent commercialization of dialogue technologies, particularly within the spoken dialogue community. We hypothesize that a particular type of cooperative response, *intensional summaries*, are effective for when users are unfamiliar with the domain. We evaluate this hypothesis with two experiments with CRUISER, a DS for in-car or mobile users to access restaurant information. First, we compare CRUISER with a baseline system-initiative DS, and show that users prefer CRUISER. Then, we experiment with four algorithms for constructing *intensional summaries* in CRUISER, and show that two summary types are equally effective: summaries that maximize domain coverage and summaries that maximize utility with respect to a user model.

## 1 Introduction

Research in cooperative response generation has had a long history and produced many useful algorithms and observations (Mays 1980; Pollack et al.1982; Joshi et al., 1986; Kalita et al., 1986) *inter alia*. However, it has had little impact on the recent commercialization of dialogue technologies, particularly within the spoken dialogue community. We believe that this lack of interest in cooperative response generation arises from two limitations of previous work: (1) There has been relatively little empirical research showing that cooperative responses lead

<i>U1:</i>	Tell me about restaurants in London.
<i>SI1:</i>	What kind of cuisine are you interested in?
<i>C1:</i>	I know of 596 restaurants in London. I know of 3 inexpensive vegetarian restaurants and 14 inexpensive Chinese restaurants. I also know of 4 inexpensive Greek restaurants.
<i>U2:</i>	Chinese.
<i>SI2:</i>	Do you have a price range in mind?
<i>C2:</i>	I know of 27 restaurants in London that serve Chinese food. There are 8 inexpensive Chinese restaurants in Chinatown and 2 inexpensive Chinese restaurants in Hampstead/Kilburn. I also know of 1 inexpensive Chinese restaurant in Soho.
<i>U3:</i>	How about a cheap one?
<i>SI3:</i>	What neighborhood would you like?
<i>C3:</i>	I know of 1 inexpensive Chinese restaurant in Hampstead/Kilburn with very good food quality and 1 in Bayswater with good food quality. I also know of 2 in Chinatown with medium food quality.

Figure 1: Intensional summaries (C = CRUISER) as compared with a *system initiative* (SI) strategy in the London restaurant domain. U = User

to more natural, effective, or efficient dialogues (Litman et al.1998; Demberg and Moore, 2006); and (2) Previous work has hand-crafted such responses, or hand-annotated the database to support them (Kaplan, 1984; Kalita et al., 1986; Cholvy, 1990; Polifroni et al., 2003; Benamara, 2004), which has made it difficult to port and scale these algorithms.

Moreover, we believe that there is an even greater need today for cooperative response generation. Larger and more complex datasets are daily being created on the Web, as information

is integrated across multiple sites and vendors. Many users will want to access this information from a mobile device and will have little knowledge of the domain. We hypothesize that these users will need cooperative responses that select and generalize the information provided.

In particular, we hypothesize that a particular type of cooperative response, *intensional summaries*, when provided incrementally during a dialogue, are effective for large or complex domains, or when users are unfamiliar with the domain. These intensional summaries have the ability to describe the data that forms the knowledge base of the system, as well as relationships among the components of that database. We have implemented *intensional summaries* in CRUISER (Cooperative Responses Using Intensional Summaries of Entities and Relations), a DS for in-car or mobile users to access restaurant information (Becker et al.2006; Weng et al.2005; Weng et al.2006). Figure 1 contrasts our proposed *intensional summary* strategy with the *system initiative* strategy used in many dialogue systems (Walker et al., 2002; VXML, 2007).

Previous research on cooperative responses has noted that summary strategies should vary according to the context (Sparck Jones, 1993), and the interests and preferences of the user (Gaasterland et al., 1992; Carenini and Moore, 2000; Demberg and Moore, 2006). A number of proposals have emphasized the importance of making generalizations (Kaplan, 1984; Kalita et al., 1986; Joshi et al., 1986). In this paper we explore different methods for constructing intensional summaries and investigate their effectiveness. We present fully automated algorithms for constructing intensional summaries using knowledge discovery techniques (Acar, 2005; Lesh and Mitzenmacher, 2004; Han et al., 1996), and decision-theoretic user models (Carenini and Moore, 2000).

We first explain in Sec. 2 our fully automated, domain-independent algorithm for constructing intensional summaries. Then we evaluate our intensional summary strategy with two experiments. First, in Sec. 3, we test the hypothesis that users prefer summary responses in dialogue

systems. We also test a refinement of that hypothesis, i.e., that users prefer summary type responses when they are unfamiliar with a domain. We compare several versions of CRUISER with the system-initiative strategy, exemplified in Fig. 1, and show that users prefer CRUISER. Then, in Sec. 4, we test four different algorithms for constructing *intensional summaries*, and show in Sec. 4.1 that two summary types are equally effective: summaries that maximize domain coverage and summaries that maximize utility with respect to a user model. We also show in Sec. 4.2 that we can predict with 68% accuracy which summary type to use, a significant improvement over the majority class baseline of 47%. We sum up in Sec. 5.

## 2 Intensional Summaries

This section describes algorithms which result in the four types of intensional summaries shown in Fig. 2. We first define *intensional summaries* as follows. Let  $D$  be a domain comprised of a set  $R$  of database records  $\{r_1, \dots, r_n\}$ . Each record consists of a set of attributes  $\{A_1, \dots, A_n\}$ , with associated values  $v$ :  $D(A_i) = \{v_{i,1}, v_{i,2}, \dots, v_{i,n}\}$ . In a dialogue system, a constraint is a value introduced by a user with either an explicit or implied associated attribute. A constraint  $c$  is a function over records in  $D$  such that  $c_j(R)$  returns a record  $r$  if  $r \subseteq D$  and  $r : A_i = c$ . The set of all dialogue constraints  $\{c_1, \dots, c_n\}$  is the *context*  $C$  at any point in the dialogue. The set of records  $R$  in  $D$  that satisfy  $C$  is the *focal information*:  $R$  is the *extension* of  $C$  in  $D$ . For example, the attribute *cuisine* in a restaurant domain has values such as “French” or “Italian”. A user utterance instantiating a constraint on cuisine, e.g., “I’m interested in Chinese food”, results in a set of records for restaurants serving Chinese food. *Intensional summaries* as shown in Fig. 2 are descriptions of the focal information, that highlight particular subsets of the focal information and make generalizations over these subsets.

The algorithm for constructing intensional summaries takes as input the focal information  $R$ , and consists of the following steps:

- Rank attributes in context  $C$ , using one of two ranking methods (Sec. 2.1);

Type	Ranking	#atts	Clusters	Scoring	Summary
<b>Ref-Sing</b>	Refiner	3	Single value	Size	<i>I know of 35 restaurants in London serving Indian food. All price ranges are represented. Some of the neighborhoods represented are Mayfair, Soho, and Chelsea. Some of the nearby tube stations are Green Park, South Kensington and Piccadilly Circus.</i>
<b>Ref-Assoc</b>	Refiner	2	Associative	Size	<i>I know of 35 restaurants in London serving Indian food. There are 3 medium-priced restaurants in Mayfair and 3 inexpensive ones in Soho. There are also 2 expensive ones in Chelsea.</i>
<b>UM-Sing</b>	User model	3	Single value	Utility	<i>I know of 35 restaurants in London serving Indian food. There are 6 with good food quality. There are also 12 inexpensive restaurants and 4 with good service quality.</i>
<b>UM-Assoc</b>	User model	2	Associative	Utility	<i>I know of 35 restaurants in London serving Indian food. There are 4 medium-priced restaurants with good food quality and 10 with medium food quality. There are also 4 that are inexpensive but have poor food quality.</i>

Figure 2: Four intensional summary types for a task specifying restaurants with Indian cuisine in London.

- Select top- $N$  attributes and construct clusters using selected attributes (Sec. 2.2);
- Score and select top- $N$  clusters (Sec. 2.3);
- Construct frames for generation, perform aggregation and generate responses.

## 2.1 Attribute Ranking

We explore two candidates for attribute ranking: User model and Refiner.

**User model:** The first algorithm utilizes decision-theoretic user models to provide an attribute ranking specific to each user (Carenini and Moore, 2000). The database contains 596 restaurants in London, with up to 19 attributes and their values. To utilize a user model, we first elicit user ranked preferences for domain attributes. Attributes that are unique across all entities, or missing for many entities, are automatically excluded, leaving six attributes: *cuisine*, *decor quality*, *food quality*, *price*, *service*, and *neighborhood*. These are ranked using the SMARTER procedure (Edwards and Barron, 1994). Rankings are converted to weights ( $w$ ) for each attribute, with a formula which guarantees that the weights sum to 1:

$$w_k = \frac{1}{K} \sum_{i=k}^K \frac{1}{i}$$

where  $K$  equals the number of attributes in the ranking. The absolute rankings are used to select attributes. The weights are also used for cluster scoring in Sec. 2.3. User model ranking is used to produce **UM-Sing** and **UM-Assoc** in Fig. 2.

**Refiner method:** The second attribute ranking method is based on the Refiner algorithm for summary construction (Polifroni et al., 2003). The Refiner returns values for every attribute in the focal information in frames ordered by frequency. If the counts for the top- $N$  (typically, 4) values for a particular attribute, e.g., *cuisine*, exceeded  $M\%$  (typically 80%) of the total counts for all values, then that attribute is selected. For example, 82% of Indian restaurants in the London database are in the neighborhoods Mayfair, Soho, and Chelsea. *Neighborhood* would, therefore, be chosen as an attribute to speak about for Indian restaurants. The thresholds  $M$  and  $N$  in the original Refiner were set *a priori*, so it was possible that no attribute met or exceeded the thresholds for a particular subset of the data. In addition, some entities could have many unknown values for some attributes.

Thus, to insure that all user queries result in some summary response, we modify the Refiner

method to include a ranking function for attributes. This function favors attributes that contain fewer unknown values but always returns a ranked set of attributes. Refiner ranking is used to produce **Ref-Sing** and **Ref-Assoc** in Fig. 2.

## 2.2 Subset Clustering

Because the focal information is typically too large to be enumerated, a second parameter attempts to find interesting clusters representing subsets of the focal information to use for the content of intensional summaries. We assume that the *coverage* of the summary is important, i.e., the larger the cluster, the more general the summary.

The simplest algorithm for producing clusters utilizes a specified number of the top-ranked attributes to define a cluster. Single attributes, as in the **Ref-Sing** and **UM-Sing** examples in Fig. 2, typically produce large clusters. Thus one algorithm uses the top three attributes to produce clusters, defined by either a single value (e.g., **UM-Sing**) or by the set of values that comprise a significant portion of the total (e.g., **Ref-Sing**).

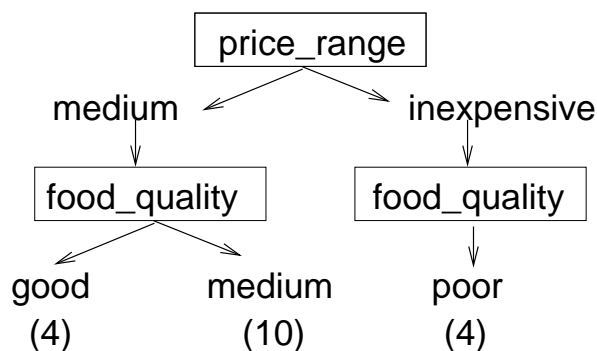


Figure 3: A partial tree for Indian restaurants in London, using price range as the predictor variable and food quality as the dependent variable. The numbers in parentheses are the size of the clusters described by the path from the root.

However, we hypothesize that more informative and useful intensional summaries might be constructed from clusters of discovered *associations* between attributes. For example, associations between *price* and *cuisine* produce summaries such as *There are 49 medium-priced*

*restaurants that serve Italian cuisine*. We apply c4.5 decision tree induction to compute associations among attributes (Kamber et al., 1997; Quinlan, 1993). Each attribute in turn is designated as the dependent variable, with other attributes used as predictors. Thus, each branch in the tree represents a cluster described by the attribute/value pairs that predict the leaf node. Fig. 3 shows clusters of different sizes induced from Indian restaurants in London. The cluster size is determined by the number of attributes used in tree induction. With two attributes, the average cluster size at the leaf node is 60.4, but drops to 4.2 with three attributes. Thus, we use two attributes to produce associative clusters, as shown in Fig. 2 (i.e., the **Ref-Assoc** and **UM-Assoc** responses), to favor larger clusters.

## 2.3 Cluster Scoring

The final parameter scores the clusters. One scoring metric is based on cluster size. Single attributes produce large clusters, while association rules produce smaller clusters.

The second scoring method selects clusters of high utility according to a user model. We first assign scalar values to the six ranked attributes (Sec. 2.1), using clustering methods as described in (Polifroni et al., 2003). The weights from the user model and the scalar values for the attributes in the user model yield an overall utility  $U$  for a cluster  $h$ , similar to utilities as calculated for individual entities (Edwards and Barron, 1994; Carenini and Moore, 2000):

$$U_h = \sum_{k=1}^K w_k(x_{hk})$$

We use cluster size scoring with Refiner ranking and utility scoring with user model ranking. For conciseness, all intensional summaries are based on the three highest scoring clusters.

## 2.4 Summary

The algorithms for attribute selection and cluster generation and scoring yield the four summary types in Table 2. Summary **Ref-Sing** is constructed using (1) the Refiner attribute ranking; and (2) no association rules. (The quantifier (e.g., *some*, *many*) is based on the cover-

age.) Summary **Ref-Assoc** is constructed using (1) the Refiner attribute ranking; and (2) association rules for clustering. Summary **UM-Sing** is constructed using (1) a user model with ranking as above; and (2) no association rules. Summary **UM-Assoc** is constructed using (1) a user model with ranking of price, food, cuisine, location, service, and decor; and (2) association rules.

### 3 Experiment One

This experiment asks whether subjects prefer intensional summaries to a baseline system-initiative strategy. We compare two types of intensional summary responses from Fig. 2, **Ref-Assoc** and **UM-Assoc** to system-initiative.

The 16 experimental subjects are asked to assume three personas, in random order, chosen to typify a range of user types, as in (Demberg and Moore, 2006). Subjects were asked to read the descriptions of each persona, which were available for reference, via a link, throughout the experiment.

The first persona is the *Londoner*, representing someone who knows London and its restaurants quite well. The Londoner persona typically knows the specific information s/he is looking for. We predict that the system-initiative strategy in Fig. 1 will be preferred by this persona, since our hypothesis is that users prefer intensional summaries when they are *unfamiliar* with the domain.

The second persona is the *Generic tourist* (GT), who doesn't know London well and does not have strong preferences when it comes to selecting a restaurant. The GT may want to *browse* the domain, i.e. to learn about the structure of the domain and retrieve information by recognition rather than specification (Belkin et al., 1994). We hypothesize that the **Ref-Assoc** strategy in Fig. 2 will best fit the GT, since the corresponding clusters have good domain coverage.

The third persona is the *UM tourist* (UMT). This persona may also want to *browse* the database, since they are unfamiliar with London. However, this user has expressed preferences about restaurants through a previous interaction. The UMT in our experiment is con-

cerned with price and food quality (in that order), and prefers restaurants in Central London. After location, the UMT is most concerned with cuisine type. The intensional summary labelled **Um-Assoc** in Fig. 2 is based on this user model, and is computed from discovered associations among preferred attributes.

As each persona, subjects rate responses on a Likert scale from 1-7, for each of four dialogues, each containing between three and four query/response pairs. We do not allow tie votes among the three choices.

#### 3.1 Experimental results

The primary hypothesis of this work is that users prefer summary responses in dialogue systems, without reference to the context. To test this hypothesis, we first compare Londoner responses (average rating 4.64) to the most highly rated of the two intensional summaries (average rating 5.29) for each query/response pair. This difference is significant ( $df = 263, p < .0001$ ), confirming that over users prefer an intensional summary strategy to a system-initiative strategy.

Table 1 shows ratings as a function of persona and response type. Overall, subjects preferred the responses tailored to their persona. The Londoner persona significantly preferred Londoner over UMT responses ( $df = 95, p < .05$ ), but not more than GT responses. This confirms our hypothesis that users prefer incremental summaries in dialogue systems. Further, it disconfirms our refinement of that hypothesis, that users prefer summaries only when they are unfamiliar with the domain. The fact that no difference was found between Londoner and GT responses indicates that GT responses contain information that is perceived as useful even when users are familiar with the domain.

The Generic Tourist persona also preferred the GT responses, significantly more than the Londoner responses ( $df = 95, p < .05$ ), but not significantly more than the UMT responses. We had hypothesized that the optimal summary type for users completely new to a domain would describe attributes that have high coverage of the focal information. This hypothesis is disconfirmed by these findings, that indicate that user

Persona	Response Type		
	London	GT	UMT
London	5.02	4.55	4.32
GT	4.14	4.67	4.39
UM tourist	3.68	4.86	5.23

Table 1: Ratings by persona assumed. London = Londoner persona, GT = Generic tourist, UMT = User Model tourist

model information is helpful when constructing summaries for any user interested in browsing.

Finally, the UM Tourist persona overwhelmingly preferred UMT responses over Londoner responses ( $df = 95, p < .0001$ ). However, UMT responses were not significantly preferred to GT responses. This confirms our hypothesis that users prefer summary responses when they are unfamiliar with the domain, but disconfirms the hypothesis that users will prefer summaries based on a user model. The results for both the Generic Tourist and the UM Tourist show that both types of intensional summaries contain useful information.

## 4 Experiment Two

The first experiment shows that users prefer intensional summaries; the purpose of the second experiment is to investigate what makes a good intensional summary. We test the different ways of constructing such summaries described in Sec. 2, and illustrated in Fig. 2.

Experimental subjects were 18 students whose user models were collected as described in Sec. 2.3. For each user, the four summary types were constructed for eight tasks in the London restaurant domain, where a task is defined by a query instantiating a particular attribute/value combination in the domain (e.g., *I'm interested in restaurants in Soho*). The tasks were selected to utilize a range of attributes. The focal information for four of the tasks (*large set tasks*) were larger than 100 entities, while the focal information for the other four tasks were smaller than 100 entities (*small set tasks*). Each task was presented to the subject on its own web page with the four intensional summaries presented as text on the web page. Each subject was asked to carefully read and rate each al-

	User model	Refiner
Association rules	3.4	2.9
Single attributes	3.0	3.4
	User model	Refiner
Small dataset	3.1	3.4
Large dataset	3.2	2.9

Table 2: User ratings showing the interaction between clustering method, attribute ranking, and dataset size in summaries.

ternative summary response on a Likert scale of 1...5 in response to the statement, *This response contains information I would find useful when choosing a restaurant*. The subjects were also asked to indicate which response they considered the best and the worst, and to provide free-text comments about each response.

### 4.1 Hypothesis Testing Results

We performed an analysis of variance with attribute ranking (user model vs. refiner), clustering method (association rules vs. single attributes), and set size (large vs. small) as independent variables and user ratings as the dependent variable. There was a main effect for set size ( $df = 1, f = 6.7, p < .01$ ), with summaries describing small datasets (3.3 average rating) rated higher than those for large datasets (3.1 average rating).

There was also a significant interaction between attribute ranking and clustering method ( $df = 1, f = 26.8, p < .001$ ). Table 2 shows ratings for the four summary types. There are no differences between the two highest rated summaries: **Ref-Sing** (average 3.4) and **UM-Assoc** (average 3.4). See Fig. 2. This suggests that discovered associations provide useful content for intensional summaries, but only for attributes ranked highly by the user model.

In addition, there was another significant interaction between ranking method and setsize ( $df = 1, f = 11.7, p < .001$ ). The ratings at the bottom of Table 2 shows that overall, users rate summaries of small datasets higher, but users rate summaries higher for large datasets when a user model is used. With small datasets, users prefer summaries that don't utilize user model information.



We also calculate the average utility for each response (Sec. 2.1) and find a strong correlation between the rating and its utility ( $p < .005$ ). When considering this correlation, it is important to remember that utility can be calculated for all responses, and there are cases where the Refiner responses have high utility scores.

## 4.2 Summary Type Prediction

Our experimental data suggest that characteristics associated with the set of restaurants being described are important, as well as utility information derived from application of a user model. The performance of a classifier in predicting summary type will indicate if trends we discovered among user judgements carry over to an automated means of selecting which response type to use in a given context.

In a final experiment, for each task, we use the highest rated summary as a class to be predicted using C4.5 (Quinlan, 1993). Thus we have 4 classes: **Ref-Sing**, **Ref-Assoc**, **UM-Sing**, and **UM-Assoc**. We derive two types of feature sets from the responses: features derived from each user model and features derived from attributes of the query/response pair itself. The five feature sets for the user model are:

- *umInfo*: 6 features for the rankings for each attribute for each user’s model, e.g. a summary whose user had rated *food quality* most highly would receive a ’5’ for the feature *food quality*;
- *avgUtility*: 4 features representing an average utility score for each alternative summary response, based on its clusters (Sec. 2.3).
- *hiUtility*: 4 features representing the highest utility score among the three clusters selected for each response;
- *loUtility*: 4 features representing the lowest utility score among the three clusters selected for each response;
- *allUtility*: 12 features consisting of the high, low, and average utility scores from the previous three feature sets.

Three feature sets are derived from the query and response pair:

- *numRests*: 4 features for the coverage of each response. For summary **Ref-Assoc** in Table 2, *numRests* is 43; for summary **UM-Assoc**, *numrests* is 53.;

Sys	Feature Sets	Acc(%)
S1	<i>allUtility</i>	47.1
S2	<i>task, numRests</i>	51.5
S3	<i>allUtility, umInfo</i>	62.3*
S4	<i>allUtility, umInfo, numRests, task</i>	63.2*
S5	<i>avgUtility, umInfo, numRests, task</i>	62.5*
S6	<i>hiUtility, umInfo, numRests, task</i>	66.9*
S7	<i>hiUtility, umInfo, numRests, task, dataset</i>	68.4*
S8	<i>loUtility, umInfo, numRests, task</i>	60.3*
S9	<i>hiUtility, umInfo</i>	64.0*

Table 3: Accuracy of feature sets for predicting preferred summary type. \* =  $p < .05$  as compared to the Baseline (S1).

- *task*: A feature for the type of constraint used to generate the focal information (e.g., *cuisine, price range*).
- *dataset*: A feature for the size of the focal information subset (i.e., *big, small*), for values greater and less than 100.

Table 3 shows the relative strengths of the two types of features on classification accuracy. The majority class baseline (System S1) is 47.1%. The S2 system uses only features associated with the query/response pair, and its accuracy (51.5%) is not significantly higher than the baseline (S3 in Table 3), and combining features from the query/response pair and the user model significantly increases accuracy in all cases. We experimented with using all the utility scores (S4), as well as with using just the average (S5), the high (S6), and the low (S8). The best performance (68.4%) is for the (S7) system combination of features.

The classification rules in Table 4 for the best system (S7) suggests some bases for users’ decisions. The first rule is very simple, simply stating that, if the highest utility value of the **Ref-Sing** response is lower than a particular threshold, then use the **UM-Assoc** response. In other words, if one of the two highest scoring response types has a low utility, use the other.

The second rule in Table 4 shows the effect that the number of restaurants in the response has on summary choice. In this rule, the **Ref-Sing** response is preferred when the highest util-

```

IF (HighestUtility: Ref-Sing) < 0.18
  THEN USE UM-Assoc
-----
IF (HighestUtility: Ref-Assoc) > 0.18) &&
  (NumRestaurants: UM-Assoc < 400) &&
  (HighestUtility: UM-Assoc < .47)
  THEN USE Ref-Sing
-----
IF (NumRestaurants: UM-Assoc < 400) &&
  (HighestUtility: UM-Assoc < .57) &&
  (HighestUtility: Ref-Assoc > .2)
  THEN USE Ref-Assoc

```

Table 4: Example classification rules from System 7 in Table 3.

ity value of that response is over a particular threshold.

The final rule in Table 4 predicts **Ref-Assoc**, the lowest overall scoring response type. When the number of restaurants accounted for by **UM-Assoc**, as well as the highest utility for that response, are both below a certain threshold, and the highest utility for the **Ref-Assoc** response is above a certain threshold, then use **Ref-Assoc**. The utility for any summary type using the Refiner method is usually lower than those using the user model, since overall utility is not taken into account in summary construction. However, even low utility summaries may mention attributes the user finds important. That, combined with higher coverage, could make that summary type preferable over one constructed to maximize user model utility.

## 5 Conclusion

We first compared intensional summary cooperative responses against a system initiative dialogue strategy in CRUISER. Subjects assumed three “personas”, a native Londoner, a tourist who was interacting with the system for the first time (GT), or a tourist for which the system has a user model (UMT). The personas were designed to reflect differing ends of the spectra defined by Belkin to characterize information-seeking strategies (Belkin et al., 1994). There was a significant preference for intensional summaries across all personas, but especially when the personas were unfamiliar with the domain.

This preference indicates that the benefits of intensional summaries outweigh the increase in verbosity.

We then tested four algorithms for summary construction. Results show that intensional summaries based on a user model with association rules, or on the Refiner method (Polifroni et al., 2003), are equally effective. While (Demberg and Moore, 2006) found that their user model stepwise refinement (UMSR) method was superior to the Refiner method, they also found many situations (70 out of 190) in which the Refiner method was preferred. Our experiment was structured differently, but it suggests that, in certain circumstances, or within certain domains, users may wish to hear about choices based on an analysis of focal information, irrespective of user preferences.

Our intensional summary algorithms automatically construct summaries from a database, along with user models collected via a domain-independent method; thus we believe that the methods described here are domain-independent. Furthermore, in tests to determine whether a classifier can predict the best summary type to use in a given context, we achieved an accuracy of 68% as compared to a majority class baseline of 47%, using dialogue context features. Both of these results point hopefully towards a different way of automating dialogue design, one based on a combination of user modelling and an analysis of contextual information. In future work we hope to test these algorithms in other domains, and show that intensional summaries can not only be automatically derived but also lead to reduced task times and increased task success.

## References

- A.C. Acar and A. Motro. 2005. Intensional Encapsulations of Database Subsets via Genetic Programming. *Proc, 16th Int. Conf. on Database and Expert Systems Applications*. Copenhagen.
- Tilman Becker, Nate Blaylock, Ciprian Gerstenberger, Ivana Kruijff-Korbayová, Andreas Korthauer, Manfred Pinkal, Michael Pitz, Peter Poller, and Jan Schehl. Natural and intuitive multimodal dialogue for in-car applications: The sammie system. In *ECAI*, pages 612–616, 2006.

- N. J. Belkin, C. Cool, A. Stein and U. Thiel. 1994. Cases, Scripts, and Information Seeking Strategies: On the Design of Interactive Information Retrieval Systems. *Expert Systems and Applications*, 9(3):379–395.
- F. Benamara. 2004. Generating Intensional Answers in Intelligent Question Answering Systems. *Proc. 3rd Int. Conf. on Natural Language Generation INLG*.
- G. Carenini and J. Moore. 2000. A Strategy for Generating Evaluative Arguments. *Proc. First Int'l Conf. on Natural Language Generation*. 1307–1314.
- Brant Cheikes and Bonnie Webber. Elements of a computational model of cooperative response generation. In *Proc. Speech and Natural Language Workshop*, pages 216–220, Philadelphia, 1989.
- X. Chen and Y-F. Wu. 2006. Personalized Knowledge Discovery: Mining Novel Association Rules from Text. *Proc., SIAM Conference on Data Mining*.
- L. Cholvy. 1990. Answering Queries Addressed to a Rule Base. *Revue d'Intelligence Artificielle*. 1(1):79–98.
- V. Demberg and J. Moore. 2006 Information Presentation in Spoken Dialogue Systems. *Proc. 11th Conf. EACL*.
- W. Edwards and F. Hutton Barron. 1994. Smarts and smarter: Improved simple methods for multiattribute utility measurement. *Organizational Behavior and Human Decision Processes*. 60:306–325.
- T. Gaasterland and P. Godfrey and J. Minker. 1992. An Overview of Cooperative Answering. *Journal of Intelligent Information Systems*. 1(2):387–416.
- J. Han, Y. Huang and N. Cercone. 1996. Intelligent Query Answering by Knowledge Discovery Techniques. *IEEE Transactions on Knowledge and Data Engineering*. 8(3):373–390.
- Aravind Joshi, Bonnie Webber, and Ralph M. Weischedel. Living up to expectations: computing expert responses. In *HLT '86: Proceedings of the workshop on Strategic computing natural language*, pages 179–189, Morristown, NJ, USA, 1986. Association for Computational Linguistics.
- J. Kalita and M.J. Colburn and G. McCalla. 1984. A response to the need for summary responses. *COLING-84*. 432–436.
- M. Kamber, L. Winstone, W. Gong, S. Cheng and J Han. 1997. Generalization and decision tree induction: efficient classification in data mining. *Proc. 7th Int. Workshop on Research Issues in Data Engineering (RIDE '97)*. 111–121.
- S.J.Kaplan. 1984. Designing a Portable Natural Language Database Query System. *ACM Transactions on Database Systems*, 9(1):1–19.
- N. Lesh and M. Mitzenmacher. Interactive data summarization: an example application. *Proc., Working Conference on Advanced Visual Interfaces*. Gallipoli, Italy. pages 183–187.
- Diane J. Litman, Shimei Pan, and Marilyn A. Walker. Evaluating response strategies in a web-based spoken dialogue agent. In *COLING-ACL*, pages 780–786, 1998.
- J. Polifroni, G. Chung, and S. Seneff. 2003. Towards the Automatic Generation of Mixed-Initiative Dialogue Systems from Web Content. *Proc. Eurospeech*. 2721–2724.
- E. Mays. Correcting misconceptions about database structure. In *Proceedings of the CSCSI '80*, 1980.
- Martha E. Pollack, Julia Hirschberg, and Bonnie L. Webber. User participation in the reasoning processes of expert systems. In *AAAI*, pages 358–361, 1982.
- J.R. Quinlan 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann. San Mateo, CA.
- K. Sparck Jones. 1998. Automatic summarising: factors and directions. I. Mani and M. Maybury, eds. *Advances in Automatic Text Summarization*. MIT Press.
- M. Walker, A. Rudnicky, J. Aberdeen, E. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, R. Passonneau, R. Prasad, S. Roukos, G. Sanders, S. Seneff and D. Stallard. 2002. DARPA Communicator Evaluation: Progress from 2000 to 2001. *Proc, ICSLP 2002*.
- F. Weng, L. Cavedon, B. Raghunathan, D. Mirkovic, H. Cheng, H. Schmidt, H. Bratt, R. Mishra, S. Peters, L. Zhao, S. Upson, E. Shriberg, and C. Bergmann. Developing a conversational dialogue system for cognitively overloaded drivers. In *Proceedings, International Congress on Intelligent Transportation Systems*, 2005.
- F. Weng, S. Varges, B. Raghunathan, F. Ratiu, H. Pon-Barry, B. Lathrop, Q. Zhang, T. Scheideck, H. Bratt, K. Xu, M. Purver, R. Mishra, M. Raya, S. Peters, Y. Meng, L. Cavedon, and L. Shriberg. Chat: A conversational helper for automotive tasks. In *Proceedings, Interspeech: International Conference on Spoken Language Processing*, 2006.
- Voxeo. VoiceXML Development Guide. <http://voicexml.org>.

# Word Clustering and Word Selection based Feature Reduction for MaxEnt based Hindi NER

**Sujan Kumar Saha**

Indian Institute of Technology  
Kharagpur, West Bengal  
India - 721302

sujan.kr.saha@gmail.com

**Pabitra Mitra**

Indian Institute of Technology  
Kharagpur, West Bengal  
India - 721302

pabitra@gmail.com

**Sudeshna Sarkar**

Indian Institute of Technology  
Kharagpur, West Bengal  
India - 721302

shudeshna@gmail.com

## Abstract

Statistical machine learning methods are employed to train a Named Entity Recognizer from annotated data. Methods like Maximum Entropy and Conditional Random Fields make use of features for the training purpose. These methods tend to overfit when the available training corpus is limited especially if the number of features is large or the number of values for a feature is large. To overcome this we proposed two techniques for feature reduction based on word clustering and selection. A number of word similarity measures are proposed for clustering words for the Named Entity Recognition task. A few corpus based statistical measures are used for important word selection. The feature reduction techniques lead to a substantial performance improvement over baseline Maximum Entropy technique.

## 1 Introduction

Named Entity Recognition (NER) involves locating and classifying the names in a text. NER is an important task, having applications in information extraction, question answering, machine translation and in most other Natural Language Processing (NLP) applications. NER systems have been developed for English and few other languages with high accuracy. These belong to two main categories based on machine learning (Bikel et al., 1997; Borthwick, 1999; McCallum and Li, 2003) and language or domain specific rules (Grishman, 1995; Wakao et al., 1996).

In English, the names are usually capitalized which is an important clue for identifying a name. Absence of capitalization makes the Hindi NER task difficult. Also, person names are more diverse in Indian languages, many common words being used as names.

A pioneering work on Hindi NER is by Li and McCallum (2003) where they used Conditional Random Fields (CRF) and feature induction to automatically construct only the features that are important for recognition. In an effort to reduce overfitting, they use a combination of a Gaussian prior and early-stopping.

In their Maximum Entropy (MaxEnt) based approach for Hindi NER development, Saha et al. (2008) also observed that the performance of the MaxEnt based model often decreases when huge number of features are used in the model. This is due to overfitting which is a serious problem in most of the NLP tasks in resource poor languages where annotated data is scarce.

This paper is a study on effectiveness of word clustering and selection as feature reduction techniques for MaxEnt based NER. For clustering we use a number of word similarities like cosine similarity among words and co-occurrence, along with the k-means clustering algorithm. The clusters are then used as features instead of words. For important word selection we use corpus based statistical measurements to find the importance of the words in the NER task. A significant performance improvement over baseline MaxEnt was observed after using the above feature reduction techniques.

The paper is organized as follows. The MaxEnt

based NER system is described in Section 2. Various approaches for word clustering are discussed in Section 3. Next section presents the procedure for selecting the important words. In Section 5 experimental results and related discussions are given. Finally Section 6 concludes the paper.

## 2 Maximum Entropy Based Model for Hindi NER

Maximum Entropy (MaxEnt) principle is a commonly used technique which provides probability of belongingness of a token to a class. MaxEnt computes the probability  $p(o|h)$  for any  $o$  from the space of all possible outcomes  $O$ , and for every  $h$  from the space of all possible histories  $H$ . In NER, history can be viewed as all information derivable from the training corpus relative to the current token. The computation of probability ( $p(o|h)$ ) of an outcome for a token in MaxEnt depends on a set of features that are helpful in making predictions about the outcome. The features may be binary-valued or multi-valued. Given a set of features and a training corpus, the MaxEnt estimation process produces a model in which every feature  $f_i$  has a weight  $\alpha_i$ . We can compute the conditional probability as (Berger et al., 1996):

$$p(o|h) = \frac{1}{Z(h)} \prod_i \alpha_i^{f_i(h,o)} \quad (1)$$

$$Z(h) = \sum_o \prod_i \alpha_i^{f_i(h,o)} \quad (2)$$

The conditional probability of the outcome is the product of the weights of all active features, normalized over the products of all the features. For our development we have used a Java based open-nlp MaxEnt toolkit<sup>1</sup>. A beam search algorithm is used to get the most probable class from the probabilities.

### 2.1 Training Corpus

The training data for the Hindi NER task is composed of about 243K words which is collected from the popular daily Hindi newspaper ‘‘Dainik Jagaran’’. This corpus has been manually annotated and contains about 16,491 Named Entities (NEs). In this study we have considered 4 types

<sup>1</sup><http://sourceforge.net/projects/maxent/>

Type	Features
Word	$w_i, w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}$
NE Tag	$t_{i-1}, t_{i-2}$
Digit information	Contains digit, Only digit, Four digit, Numerical word
Affix information	Fixed length suffix, Suffix list, Fixed length prefix
POS information	POS of words, Coarse-grained POS, POS based binary features

Table 1: Features used in the MaxEnt based Hindi NER system

of NEs, these are *Person* (Per), *Location* (Loc), *Organization* (Org) and *Date* (Dat). To recognize entity boundaries each name class  $N$  has 4 types of labels:  $N\_Begin$ ,  $N\_Continue$ ,  $N\_End$  and  $N\_Unique$ . For example, *Kharagpur* is annotated as *Loc\_Unique* and *Atal Bihari Vajpeyi* is annotated as *Per\_Begin Per\_Continue Per\_End*. Hence, there are a total of 17 classes including one class for not-name. The corpus contains 6298 person, 4696 location, 3652 organization and 1845 date entities.

### 2.2 Feature Description

We have identified a number of candidate features for the Hindi NER task. Several experiments were conducted with the identified features, individually and in combination. Some of the features are mentioned below. They are summarized in Table 1.

**Static Word Feature:** Recognition of NE is highly dependent on contexts. So the surrounding words of a particular word ( $w_i$ ) are used as features. During our experiments different combinations of previous 3 words ( $w_{i-3}...w_{i-1}$ ) to next 3 words ( $w_{i+1}...w_{i+3}$ ) are treated as features. This is represented by  $L$  binary features where  $L$  is the size of lexicon.

**Dynamic NE tag:** NE tags of the previous words ( $t_{i-m}...t_{i-1}$ ) are used as features. During decoding, the value of this feature for a word ( $w_i$ ) is obtained only after the computation of the NE tag for the previous word ( $w_{i-1}$ ).

**Digit Information:** If a word ( $w_i$ ) contains digit(s) then the feature *ContainsDigit* is set to 1. This feature is used with some modifications also. *OnlyDigit*, which is set to 1 if the word contains

Feature Id	Feature	Per	Loc	Org	Dat	Total
F1	$w_i, w_{i-1}, w_{i+1}$	61.36	68.29	52.12	88.9	67.26
F2	$w_i, w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}$	64.10	67.81	58	92.30	69.09
F3	$w_i, w_{i-1}, w_{i-2}, w_{i-3}, w_{i+1}, w_{i+2}, w_{i+3}$	60.42	67.81	51.48	90.18	66.84
F4	$w_i, w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}, t_{i-1}, t_{i-2}, \text{Suffix}$	66.67	73.36	58.58	89.09	71.2
F5	$w_i, w_{i-1}, w_{i+1}, t_{i-1}, \text{Suffix}$	69.65	75.8	59.31	89.09	73.42
F6	$w_i, w_{i-1}, w_{i+1}, t_{i-1}, \text{Prefix}$	66.67	71	58.58	87.8	70.02
F7	$w_i, w_{i-1}, w_{i+1}, t_{i-1}, \text{Prefix}, \text{Suffix}$	70.61	71	59.31	89.09	72.5
F8	$w_i, w_{i-1}, w_{i+1}, t_{i-1}, \text{Suffix}, \text{Digit}$	70.61	75.8	60.54	93.8	74.26
F9	$w_i, w_{i-1}, w_{i+1}, t_{i-1}, \text{POS (28 tags)}$	64.25	71	60.54	89.09	70.39
F10	$w_i, w_{i-1}, w_{i+1}, t_{i-1}, \text{POS (coarse grained)}$	69.65	75.8	59.31	92.82	74.16
F11	$w_i, w_{i-1}, w_{i+1}, T_{i-1}, \text{Suffix}, \text{Digit}, \text{NomPSP}$	72.26	78.6	61.36	92.82	75.6
F12	$w_i, w_{i-1}, w_{i+1}, w_{i-2}, w_{i+2}, T_{i-1}, \text{Prefix}, \text{Suffix}, \text{Digit}, \text{NomPSP}$	65.26	78.01	52.12	93.33	72.65

Table 2: F-values for different features in the MaxEnt based Hindi NER system

only digits, *4Digit*, which is set to 1 if the word contains only 4 digits, etc. are some modifications of the feature which are helpful.

**Numerical Word:** For a word ( $w_i$ ) if it is a numerical word i.e. word denoting a number (e.g. *eka*<sup>2</sup> (one), *do* (two), *tina* (three) etc.) then the feature *NumWord* is set to 1.

**Word Suffix:** Word suffix information is helpful to identify the NEs. Two types of suffix features have been used. Firstly a fixed length word suffix (set of characters occurring at the end of the word) of the current and surrounding words are used as features. Secondly we compiled list of common suffixes of place names in Hindi. For example, *pura*, *bAda*, *nagara* etc. are location suffixes. We used binary feature corresponding to the list - whether a given word has a suffix from the list.

**Word Prefix:** Prefix information of a word may be also helpful in identifying whether it is a NE. A

fixed length word prefix (set of characters occurring at the beginning of the word) of current and surrounding words are treated as features. List of important prefixes, which are used frequently in the NEs, are also effective.

**Parts-of-Speech (POS) Information:** The POS of the current word and the surrounding words are used as feature for NER. We have used a Hindi POS tagger developed at IIT Kharagpur, India which has an accuracy about 90%. We have used the POS values of the current and surrounding words as features.

We realized that the detailed POS tagging is not very relevant. Since NEs are noun phrases, the noun tag is very relevant. Further the postposition following a name may give a clue to the NE type. So we decided to use a coarse-grained tagset with only three tags - nominal (Nom), postposition (PSP) and other (O).

The POS information is also used by defining several binary features. An example is the *NomPSP* binary feature. The value of this feature is defined to be 1 if the current word is nominal and the next

<sup>2</sup>All Hindi words are written in italics using the ‘Itrans’ transliteration.

word is a PSP.

### 2.3 Performance of Hindi NER using MaxEnt Method

The performance of the MaxEnt based Hindi NER using the above mentioned features is reported here as a baseline. We have evaluated the system using a blind test corpus of 25K words. The test corpus contains 521 person, 728 location, 262 organization and 236 date entities. The accuracies are measured in terms of the f-measure, which is the weighted harmonic mean of precision and recall. Precision is the fraction of the correct annotations and recall is the fraction of the total NEs that are successfully annotated. The general formula for measuring the f-measure or f-value is,  $F_{\beta} = (1 + \beta^2) \cdot (\text{precision} \cdot \text{recall}) \setminus (\beta^2 \cdot \text{precision} + \text{recall})$ . Here the value of  $\beta$  is taken as 1. In Table 2 we have shown the accuracy values for few feature sets.

While experimenting with static word features, we have observed that a window of previous and next two words ( $w_{i-2} \dots w_{i+2}$ ) gives best result (69.09) using the word features only. But when  $w_{i-3}$  and  $w_{i+3}$  are added with it, the f-value is reduced to 66.84. Again when  $w_{i-2}$  and  $w_{i+2}$  are deducted from the feature set (i.e. only  $w_{i-1}$  and  $w_{i+1}$  as feature), the f-value is reduced to 67.26. This demonstrates that  $w_{i-2}$  and  $w_{i+2}$  are helpful features in NE identification.

When suffix, prefix and digit information are added to the feature set, the f-value is increased upto 74.26. The value is obtained using the feature set F8 [ $w_i$ ,  $w_{i-1}$ ,  $w_{i+1}$ ,  $t_{i-1}$ , Suffix, Digit]. It is observed that when  $w_{i-2}$  and  $w_{i+2}$  are added with the feature, the accuracy decreases by 2%. It contradicts the results using the word features only. Another interesting observation is that prefix information are helpful features in NE identification as these increase accuracy when separately added with the word features (F6). Similarly the suffix information helps in increasing the accuracy. But when both the suffix and prefix information are used in combination along with the word features, the f-value is decreased. From Table 2, a f-value of 73.42 is obtained using F5 [ $w_i$ ,  $w_{i-1}$ ,  $w_{i+1}$ ,  $t_{i-1}$ , Suffix] but when prefix information are added with it (F7), the f-value is reduced to 72.5.

POS information are important features in NER. In general it is observed that coarse grained POS information performs better than the finer grained POS information. The best accuracy (75.6 f-value) of the baseline system is obtained using the binary NomPSP feature along with word feature ( $w_{i-1}$ ,  $w_{i+1}$ ), suffix and digit information. It is noted that when  $w_{i-2}$ ,  $w_{i+2}$  and prefix information are added with the best feature, the f-value is reduced to 72.65.

From the above discussion it is clear that the system suffers from overfitting if a large number of features are used to train the system. Note that the surrounding word ( $w_{i-2}$ ,  $w_{i-1}$ ,  $w_{i+1}$ ,  $w_{i+2}$  etc.) features can take any value from the lexicon and hence are of high dimensionality. These cause the degradation of performance of the system. However it is obvious that few words in the lexicon are important in identification of NEs.

To solve the problem of high dimensionality we use clustering to group the words present in the corpus into much smaller number of clusters. Then the word clusters are used as features instead of the word features (for surrounding words). For example, our Hindi corpus contains 17,456 different words, which are grouped into  $N$  (say 100) clusters. Then for a particular word, it is assigned to a cluster and the corresponding cluster-id is used as feature. Hence the number of features is reduced to 100 instead of 17,456.

Similarly, selection of important words can also solve the problem of high dimensionality. As some of the words in the lexicon play important role in the NE identification process, we aim to select these particular words. Only these important words are used in NE identification instead of all words in the corpus.

## 3 Word Clustering

Clustering is the process of grouping together objects based on their similarity. The measure of similarity is critical for good quality clustering. We have experimented with some approaches to compute word-word similarity. These are described in details in the following section.

### 3.1 Cosine Similarity based on Sentence Level Co-occurrence

A word is represented by a binary vector of dimension same as the number of sentences in the corpus. A component of the vector is 1 if the word occurs in the corresponding sentence and zero otherwise. Then we measure cosine similarity between the word vectors. The cosine similarity between two word vectors ( $\vec{A}$  and  $\vec{B}$ ) with dimension  $d$  is measured as:

$$\text{CosSim}(\vec{A}, \vec{B}) = \frac{\sum_d A_d B_d}{(\sum_d A_d^2)^{\frac{1}{2}} \times (\sum_d B_d^2)^{\frac{1}{2}}} \quad (3)$$

This measures the number of co-occurring sentences.

### 3.2 Cosine Similarity based on Proximal Words

In this measure a word is represented by a vector having dimension same as the lexicon size. For ease of implementation we have taken a dimension of  $2 \times 200$ , where each component of the vector corresponds to one of the 200 most frequent preceding and following words of a token word. *List\_Prev* containing the most frequent (top 200) previous words ( $w_{i-1}$  or  $w_{i-2}$  if  $w_i$  is the first word of a NE) and *List\_Next* contains 200 most frequent next words ( $w_{i+1}$  or  $w_{i+2}$  if  $w_i$  is the last word of a NE). A particular word  $w_k$  may occur several times (say  $n$ ) in the corpus. For each occurrence of  $w_k$  find if its previous word ( $w_{k-1}$  or  $w_{k-2}$ ) matches any element of *List\_Prev*. If matches, then set 1 to the corresponding position of the vector and set zero to all other positions related to *List\_Prev*. Similarly check the next word ( $w_{k+1}$  or  $w_{k+2}$ ) in the *List\_Next* and find the values of the corresponding positions. The final word vector  $\vec{W}_k$  is obtained by taking the average of all occurrences of  $w_k$ . Then the cosine similarity is measured between the word vectors. This measures the similarity of the contexts of the occurrences of the word in terms of the proximal words.

### 3.3 Similarity based on Proximity to NE Categories

Here, for each word ( $w_i$ ) in the corpus four binary vectors are defined corresponding to two preceding

and two following positions (i-1, i-2, i+1, i+2). Each binary vector is of dimension five corresponding to four NE classes ( $C_j$ ) and one for the not-name class. For a particular word  $w_k$ , find all the words occur in a particular position (say, +1). Measure the fraction ( $P_j(w_k)$ ) of these words belonging to a class  $C_j$ . The component of the word vector  $\vec{W}_k$  for the position corresponding to  $C_j$  is  $P_j(w_k)$ .

$$P_j(w_k) = \frac{\text{No. of times } w_{k+1} \text{ is a NE of class } C_j}{\text{Total occurrence of } w_k \text{ in corpus}}$$

The Euclidean distance is used to find the similarity between the above word vectors as a similarity measure. Some of the word vectors for the +1 position are given in Table 3. In this table we have given the word vectors for a few Hindi words, which are, *sthita* (located), *shahara* (city), *jAkara* (go), *nagara* (township), *gA.nva* (village), *nivAsI* (resident), *mishrA* (a surname) and *limiTeDa* (ltd.). From the table we observe that the word vectors are close for *sthita* [0 0.478 0 0 0.522], *shahara* [0 0.585 0.001 0.024 0.39], *nagara* [0 0.507 0.019 0 0.474] and *gA.nva* [0 0.551 0 0 0.449]. So these words are considered as close.

Word	Per	Loc	Org	Dat	Not
<i>sthita</i>	0	0.478	0	0	0.522
<i>shahara</i>	0	0.585	0.001	0.024	0.39
<i>jAkara</i>	0	0.22	0	0	0.88
<i>nagara</i>	0	0.507	0.019	0	0.474
<i>gA.nva</i>	0	0.551	0	0	0.449
<i>nivAsI</i>	0.108	0.622	0	0	0.27
<i>mishrA</i>	0.889	0	0	0	0.111
<i>limiTeDa</i>	0	0	1	0	0

Table 3: Example of some word vectors for next (+1) position (see text for glosses)

### 3.4 K-means Clustering

Using the above similarity measures we have used the k-means algorithm. The seeds were randomly selected. The value of  $k$  (number of clusters) was varied till the best result is obtained.

## 4 Important Word Selection

It is noted that not all words are equally important in determining the NE category. Some of the words



in the lexicon are typically associated with a particular NE category and hence have important role to play in the classification process. We describe below a few statistical techniques that has been used to identify the important words.

#### 4.1 Class Independent Important Word Selection

We define context words as those which occur in proximity of a NE. In other words, context words are the words present in the  $w_{i-2}$ ,  $w_{i-1}$ ,  $w_{i+1}$  or  $w_{i+2}$  position if  $w_i$  is a NE. Note that only a subset of the lexicon are context words. For all the context words, its  $N\_weight$  is calculated as the ratio between the occurrence of the word as a context word and its total number of occurrence in the corpus. The context words having the higher  $N\_weight$  are considered as important words for NER. For our experiments we have considered top 500 words as important words.

$$N\_weight(w_i) = \frac{\text{Occurrence of } w_i \text{ as context word}}{\text{Total occurrence of } w_i \text{ in corpus}}$$

#### 4.2 Important Words for Each Class

Similar to the class independent important word selection from the contexts, important words are selected for individual classes also. This is an extension of the previous context word considering only NEs of a particular class. For person, location, organization and date classes we have considered top 150, 120, 50 and 50 words respectively as important words. Four binary features are also defined for these four classes. These are defined as having value 1 if any of the context words belongs to the important words list for a particular class.

#### 4.3 Important Words for Each Position

Position based important words are also selected from the corpus. Here instead of context, particular positions are considered. Four lists are compiled for two preceding and two following positions (-2, -1, +1 and +2).

### 5 Evaluation of NE Recognition

The following subsections contain the experimental results using word clustering and important word selection. The results demonstrate the effectiveness of

<b>k</b>	<b>Per</b>	<b>Loc</b>	<b>Org</b>	<b>Dat</b>	<b>Total</b>
20	66.33	74.57	43.64	91.30	69.54
50	64.13	76.35	52	93.62	71.7
80	66.33	74.57	53.85	93.62	72.08
100	70.1	73.1	57.7	96.62	<b>72.78</b>
120	66.15	73.43	54.9	93.62	71.52
150	66.88	74.94	53.06	95.65	72.33
200	66.09	73.82	52	92	71.13

Table 4: Variation of MaxEnt based system accuracy depending on number of clusters ( $k$ )

word clustering and important word selection over the baseline MaxEnt model.

#### 5.1 Using Word Clusters

To evaluate the effectiveness of the clustering approaches in Hindi NER, we have used cluster features instead of word features. For the surrounding words, corresponding cluster-ids are used as feature.

**Choice of  $k$  :** We have already mentioned that, for k-means clustering number of classes ( $k$ ) should be determined initially. To find suitable  $k$  we had conducted the following experiments. We have selected a feature F1 (mentioned in Table 2) and applied the clusters with different  $k$  as features replacing the word features. In Table 4 we have summarized the experimental results, in order to find a suitable  $k$  for clustering, the word vectors obtained using the procedure described in Section 3.3. From the table we observe that the best result is obtained when  $k$  is 100. We have used  $k = 100$  for the subsequent experiments for comparing the effectiveness of the features. Similarly when we deal with all the words in the corpus (17,465 words), we got best results when the words are clustered into 1100 clusters.  $\diamond$

The details of the comparison between the baseline word features and the reduced features obtained using clustering are given in Table 5. In general it is observed that clustering has improved the performance over baseline features. Using only cluster features the system provides a maximum f-value of 74.26 where the corresponding word features give f-value of 69.09.

Among the various similarity measures of clustering, improved results are obtained using the clus-

Feature	Using Word Features	Using Clusters (C1)	Using Clusters (C2)	Using Clusters (C3)
$w_i$ , window(-1, +1)	67.26	69.67	72.05	72.78
$w_i$ , window(-2, +2)	69.09	71.52	72.65	74.26
$w_i$ , window(-1, +1), Suffix	73.42	74.24	75.44	75.84
$w_i$ , window(-1, +1), Prefix, Suffix	72.5	74.76	75.7	76.33
$w_i$ , window(-1, +1), Prefix, Suffix, Digit	74.26	75.09	75.91	76.41
$w_i$ , window(-1, +1), Prefix, Suffix, Digit, NomPSP	75.6	77.2	77.39	77.61
$w_i$ , window(-2, +2), Prefix, Suffix, Digit, NomPSP	72.65	77.86	78.61	<b>79.03</b>

Table 5: F-values for different features in a MaxEnt based Hindi NER with clustering based feature reduction [ $window(-m, +n)$  refers to the cluster or word features corresponding to previous  $m$  positions and next  $n$  positions; C1 is the clusters which use sentence level co-occurrence based cosine similarity (3.1), C2 denotes the clusters which use proximal word based cosine similarity (3.2), C3 denotes the clusters for each positions related to NE (3.3)]

ters which uses the similarity measurement based on proximity of the words to NE categories (defined in Section 3.3).

Using clustering features the best f-value (79.03) is obtained using clusters for previous two and next two words along with the suffix, prefix, digit and POS information.

It is observed that the prefix information increases the accuracy if applied along with suffix information when cluster features are used. More interestingly, addition of cluster features for positions  $-2$  and  $+2$  over the feature [window(-1, +1), Suffix, Prefix, Digit, NomPSP] increase the f-value from 77.61 to 79.03. But in the baseline system addition of word features ( $w_{i-2}$  and  $w_{i+2}$ ) over the same feature decrease the f-value from 75.6 to 72.65.

## 5.2 Using Important Word Selection

The details of the comparison between the word feature and the reduced features based on important word selection are given in Table 6. For the surrounding word features, find whether the particular word (e.g. at position -1, -2 etc.) presents in the important words list (corresponding to the particular position if position based important words are considered). If the word occurs in the list then the word is used as features. In general it is observed that word selection also improves performance over baseline features. Among the different approaches,

the best result is obtained when important words for two preceding and two following positions (defined in Section 4.3) are selected. Using important word based features, the highest f-value of 79.85 is obtained by using the important words for previous two and next two positions along with the suffix, prefix, digit and POS information.

## 5.3 Relative Effectiveness of Clustering and Word Selection

In most of the cases clustering based features perform better than the important word based feature reduction. But the best f-value (79.85) of the system (using the clustering based and important word based features separately) is obtained by using important word based features.

Next we have made an experiment by considering both the clusters and important words combined. We have defined the combined feature as, if the word ( $w_i$ ) is in the corresponding important word list then the word is used as feature otherwise the corresponding cluster-id (in which  $w_i$  belongs to) is considered as feature. Using the combined feature, we have achieved further improvement. Here we are able to achieve the highest f-value of 80.01.

## 6 Conclusion

A hierarchical word clustering technique, where clusters are driven automatically from large unan-

Feature	Using Word Features	Using Words (I1)	Using Words (I2)	Using Words (I3)
$w_i$ , window(-1, +1)	67.26	66.31	67.53	66.8
$w_i$ , window(-2, +2)	69.09	72.04	72.9	73.34
$w_i$ , window(-1, +1), Suffix	73.42	73.85	73.12	74.61
$w_i$ , window(-1, +1), Prefix, Suffix	72.5	73.52	73.94	74.87
$w_i$ , window(-1, +1), Prefix, Suffix, Digit	74.26	73.97	74.13	74.7
$w_i$ , window(-1, +1), Prefix, Suffix, Digit, NomPSP	75.6	75.84	76.6	77.22
$w_i$ , window(-2, +2), Prefix, Suffix, Digit, NomPSP	72.65	76.69	77.42	<b>79.85</b>

Table 6: F-values for different features in a MaxEnt based Hindi NER with important word based feature reduction [ $window(-m, +n)$  refers to the important word or baseline word features corresponding to previous  $m$  positions and next  $n$  positions; I1 is the class independent important words (4.1), I2 denotes the important words for each class (4.2), I3 denotes the important words for each positions (4.3)]

notated corpus, is used by Miller et al. (2004) for augmenting annotated training data. Note that our clustering approach is different, where the clusters are obtained using some statistics derived from the annotated corpus, and also the purpose is different as we have used the clusters for feature reduction.

In this paper we propose two feature reduction techniques for Hindi NER based on word clustering and word selection. A number of word similarity measures are used for clustering. A few statistical approaches are used for the selection of important words. It is observed that significant enhancement of accuracy over the baseline system which use word features is obtained. This is probably due to reduction of overfitting. This is more important for a resource poor languages like Hindi where there is scarcity in annotated training data and other NER resources (like, gazetteer lists).

## 7 Acknowledgement

The work is partially funded by Microsoft Research India.

## References

Berger A L, Pietra S D and Pietra V D 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistic*, 22(1):39–71.  
 Bikel D M, Miller S, Schwartz R and W Ralph. 1997. Nymble: A High Performance Learning Name-finder.

In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201.  
 Borthwick A. 1999. A Maximum Entropy Approach to Named Entity Recognition. *Ph.D. thesis, Computer Science Department, New York University*.  
 Grishman R. 1995. The New York University System MUC-6 or Where’s the syntax? In *Proceedings of the Sixth Message Understanding Conference*.  
 Li W and McCallum A. 2003. Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3):290–294.  
 McCallum A and Li W. 2003. Early Results for Named Entity Recognition with Conditional Random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*.  
 Miller S, Guinness J and Zamanian A. 2004. Name Tagging with Word Clusters and Discriminative Training. In *Proceedings of the HLT-NAACL 2004*, pages 337–342.  
 Saha S K, Sarkar S and Mitra P. 2008. A Hybrid Feature Set based Maximum Entropy Hindi Named Entity Recognition. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP-08)*, pages 343–349.  
 Wakao T, Gaizauskas R and Wilks Y. 1996. Evaluation of an algorithm for the recognition and classification of proper names. In *Proceedings of COLING-96*.

# Combining EM Training and the MDL Principle for an Automatic Verb Classification incorporating Selectional Preferences

Sabine Schulte im Walde, Christian Hying, Christian Scheible, Helmut Schmid

Institute for Natural Language Processing

University of Stuttgart, Germany

{schulte,hyingcn,scheibcn,schmid}@ims.uni-stuttgart.de

## Abstract

This paper presents an innovative, complex approach to semantic verb classification that relies on selectional preferences as verb properties. The probabilistic verb class model underlying the semantic classes is trained by a combination of the EM algorithm and the MDL principle, providing soft clusters with two dimensions (verb senses and subcategorisation frames with selectional preferences) as a result. A language-model-based evaluation shows that after 10 training iterations the verb class model results are above the baseline results.

## 1 Introduction

In recent years, the computational linguistics community has developed an impressive number of semantic verb classifications, i.e., classifications that generalise over verbs according to their semantic properties. Intuitive examples of such classifications are the MOTION WITH A VEHICLE class, including verbs such as *drive*, *fly*, *row*, etc., or the BREAK A SOLID SURFACE WITH AN INSTRUMENT class, including verbs such as *break*, *crush*, *fracture*, *smash*, etc. Semantic verb classifications are of great interest to computational linguistics, specifically regarding the pervasive problem of data sparseness in the processing of natural language. Up to now, such classifications have been used in applications such as word sense disambiguation (Dorr and Jones, 1996; Kohomban and Lee, 2005), machine translation (Prescher et al., 2000; Koehn and Hoang, 2007), document classification (Klavans and Kan, 1998), and in statistical lexical acquisition in general (Rooth et al., 1999; Merlo and Stevenson, 2001; Korhonen, 2002; Schulte im Walde, 2006).

Given that the creation of semantic verb classifications is not an end task in itself, but depends on the application scenario of the classification, we find various approaches to an automatic induction of semantic verb classifications. For example, Siegel and McKeown (2000) used several machine learning algorithms to perform an automatic aspectual classification of English verbs into event and stative verbs. Merlo and Stevenson (2001) presented an automatic classification of three types of English intransitive verbs, based on argument structure and heuristics to thematic relations. Pereira et al. (1993) and Rooth et al. (1999) relied on the Expectation-Maximisation algorithm to induce soft clusters of verbs, based on the verbs' direct object nouns. Similarly, Korhonen et al. (2003) relied on the Information Bottleneck (Tishby et al., 1999) and subcategorisation frame types to induce soft verb clusters.

This paper presents an innovative, complex approach to semantic verb classes that relies on selectional preferences as verb properties. The underlying linguistic assumption for this verb class model is that verbs which agree on their selectional preferences belong to a common semantic class. The model is implemented as a soft-clustering approach, in order to capture the polysemy of the verbs. The training procedure uses the Expectation-Maximisation (EM) algorithm (Baum, 1972) to iteratively improve the probabilistic parameters of the model, and applies the Minimum Description Length (MDL) principle (Rissanen, 1978) to induce WordNet-based selectional preferences for arguments within subcategorisation frames. Our model is potentially useful for lexical induction (e.g., verb senses, subcategorisation and selectional preferences, collocations, and verb alternations),

and for NLP applications in sparse data situations. In this paper, we provide an evaluation based on a language model.

The remainder of the paper is organised as follows. Section 2 introduces our probabilistic verb class model, the EM training, and how we incorporate the MDL principle. Section 3 describes the clustering experiments, including the experimental setup, the evaluation, and the results. Section 4 reports on related work, before we close with a summary and outlook in Section 5.

## 2 Verb Class Model

### 2.1 Probabilistic Model

This paper suggests a probabilistic model of verb classes that groups verbs into clusters with similar subcategorisation frames and selectional preferences. Verbs may be assigned to several clusters (soft clustering) which allows the model to describe the subcategorisation properties of several verb readings separately. The number of clusters is defined in advance, but the assignment of the verbs to the clusters is learnt during training. It is assumed that all verb readings belonging to one cluster have similar subcategorisation and selectional properties. The selectional preferences are expressed in terms of semantic concepts from WordNet, rather than a set of individual words. Finally, the model assumes that the different arguments are mutually independent for all subcategorisation frames of a cluster. From the last assumption, it follows that any statistical dependency between the arguments of a verb has to be explained by multiple readings.

The statistical model is characterised by the following equation which defines the probability of a verb  $v$  with a subcategorisation frame  $f$  and arguments  $a_1, \dots, a_{n_f}$ :

$$p(v, f, a_1, \dots, a_{n_f}) = \sum_c p(c) p(v|c) p(f|c) * \prod_{i=1}^{n_f} \sum_{r \in R} p(r|c, f, i) p(a_i|r)$$

The model describes a stochastic process which generates a verb-argument tuple like  $\langle \textit{speak}, \textit{subj-pp.to}, \textit{professor}, \textit{audience} \rangle$  by

1. selecting some cluster  $c$ , e.g.  $c_3$  (which might

correspond to a set of *communication* verbs), with probability  $p(c_3)$ ,

2. selecting a verb  $v$ , here the verb *speak*, from cluster  $c_3$  with probability  $p(\textit{speak}|c_3)$ ,
3. selecting a subcategorisation frame  $f$ , here *subj-pp.to*, with probability  $p(\textit{subj-pp.to}|c_3)$ ; note that the frame probability only depends on the cluster, and not on the verb,
4. selecting a WordNet concept  $r$  for each argument slot, e.g. *person* for the first slot with probability  $p(\textit{person}|c_3, \textit{subj-pp.to}, 1)$  and *social group* for the second slot with probability  $p(\textit{social group}|c_3, \textit{subj-pp.to}, 2)$ ,
5. selecting a word  $a_i$  to instantiate each concept as argument  $i$ ; in our example, we might choose *professor* for *person* with probability  $p(\textit{professor}|\textit{person})$  and *audience* for *social group* with probability  $p(\textit{audience}|\textit{social group})$ .

The model contains two *hidden variables*, namely the clusters  $c$  and the selectional preferences  $r$ . In order to obtain the overall probability of a given verb-argument tuple, we have to sum over all possible values of these hidden variables.

The assumption that the arguments are independent of the verb given the cluster is essential for obtaining a clustering algorithm because it forces the EM algorithm to make the verbs within a cluster as similar as possible.<sup>1</sup> The assumption that the different arguments of a verb are mutually independent is important to reduce the parameter set to a tractable size

The fact that verbs select for concepts rather than individual words also reduces the number of parameters and helps to avoid sparse data problems. The application of the MDL principle guarantees that no important information is lost.

The probabilities  $p(r|c, f, i)$  and  $p(a|r)$  mentioned above are not represented as atomic entities. Instead, we follow an approach by Abney

<sup>1</sup>The EM algorithm adjusts the model parameters in such a way that the probability assigned to the training tuples is maximised. Given the model constraints, the data probability can only be maximised by making the verbs within a cluster as similar to each other as possible, regarding the required arguments.

and Light (1999) and turn WordNet into a Hidden Markov model (HMM). We create a new pseudo-concept for each WordNet noun and add it as a hyponym to each synset containing this word. In addition, we assign a probability to each hypernym-hyponym transition, such that the probabilities of the hyponymy links of a synset sum up to 1. The pseudo-concept nodes emit the respective word with a probability of 1, whereas the regular concept nodes are non-emitting nodes. The probability of a path in this (a priori) WordNet HMM is the product of the probabilities of the transitions within the path. The probability  $p(a|r)$  is then defined as the sum of the probabilities of all paths from the concept  $r$  to the word  $a$ . Similarly, we create a partial WordNet HMM for each argument slot  $\langle c, f, i \rangle$  which encodes the selectional preferences. It contains only the WordNet concepts that the slot selects for, according to the MDL principle (cf. Section 2.3), and the dominating concepts. The probability  $p(r|c, f, i)$  is the total probability of all paths from the top-most WordNet concept  $entity$  to the terminal node  $r$ .

## 2.2 EM Training

The model is trained on verb-argument tuples of the form described above, i.e., consisting of a verb and a subcategorisation frame, plus the nominal<sup>2</sup> heads of the arguments. The tuples may be extracted from parsed data, or from a treebank. Because of the hidden variables, the model is trained iteratively with the Expectation-Maximisation algorithm (Baum, 1972). The parameters are randomly initialised and then re-estimated with the Inside-Outside algorithm (Lari and Young, 1990) which is an instance of the EM algorithm for training Probabilistic Context-Free Grammars (PCFGs).

The PCFG training algorithm is applicable here because we can define a PCFG for each of our models which generates the same verb-argument tuples with the same probability. The PCFG is defined as follows:

- (1) The start symbol is TOP.
- (2) For each cluster  $c$ , we add a rule  $TOP \rightarrow V_c A_c$  whose probability is  $p(c)$ .

<sup>2</sup>Arguments with lexical heads other than nouns (e.g., subcategorised clauses) are not included in the selectional preference induction.

- (3) For each verb  $v$  in cluster  $c$ , we add a rule  $V_c \rightarrow v$  with probability  $p(v|c)$ .
- (4) For each subcategorisation frame  $f$  of cluster  $c$  with length  $n$ , we add a rule  $A_c \rightarrow f R_{c,f,1,entity} \dots R_{c,f,n,entity}$  with probability  $p(f|c)$ .
- (5) For each transition from a node  $r$  to a node  $r'$  in the selectional preference model for slot  $i$  of the subcategorisation frame  $f$  of cluster  $c$ , we add a rule  $R_{c,f,i,r} \rightarrow R_{c,f,i,r'}$  whose probability is the transition probability from  $r$  to  $r'$  in the respective WordNet-HMM.
- (6) For each terminal node  $r$  in the selectional preference model, we add a rule  $R_{c,f,i,r} \rightarrow R_r$  whose probability is 1. With this rule, we “jump” from the selectional restriction model to the corresponding node in the a priori model.
- (7) For each transition from a node  $r$  to a node  $r'$  in the a priori model, we add a rule  $R_r \rightarrow R_{r'}$  whose probability is the transition probability from  $r$  to  $r'$  in the a priori WordNet-HMM.
- (8) For each word node  $a$  in the a priori model, we add a rule  $R_a \rightarrow a$  whose probability is 1.

Based on the above definitions, a partial “parse” for  $\langle speak\ subj\text{-}pp.\text{to}\ professor\ audience \rangle$ , referring to cluster 3 and one possible WordNet path, is shown in Figure 1. The connections within  $R_3$  ( $R_{3,\dots,entity} - R_{3,\dots,person/group}$ ) and within  $R$  ( $R_{person/group} - R_{professor/audience}$ ) refer to sequential applications of rule types (5) and (7), respectively.

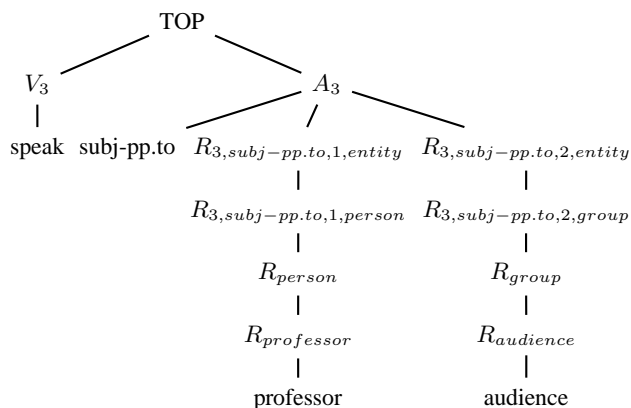


Figure 1: Example parse tree.

The EM training algorithm maximises the likelihood of the training data.

## 2.3 MDL Principle

A model with a large number of fine-grained concepts as selectional preferences assigns a higher likelihood to the data than a model with a small number of general concepts, because in general a larger number of parameters is better in describing training data. Consequently, the EM algorithm a priori prefers fine-grained concepts but – due to sparse data problems – tends to overfit the training data. In order to find selectional preferences with an appropriate granularity, we apply the Minimum Description Length principle, an approach from Information Theory. According to the MDL principle, the model with minimal *description length* should be chosen. The description length itself is the sum of the *model length* and the *data length*, with the model length defined as the number of bits needed to encode the model and its parameters, and the data length defined as the number of bits required to encode the training data with the given model. According to coding theory, an optimal encoding uses  $-\log_2 p$  bits, on average, to encode data whose probability is  $p$ . Usually, the model length increases and the data length decreases as more parameters are added to a model. The MDL principle finds a compromise between the size of the model and the accuracy of the data description.

Our selectional preference model relies on Li and Abe (1998), applying the MDL principle to determine selectional preferences of verbs and their arguments, by means of a concept hierarchy ordered by hypernym/hyponym relations. Given a set of nouns within a specific argument slot as a sample, the approach finds the cut<sup>3</sup> in a concept hierarchy which minimises the sum of encoding both the model and the data. The *model length* ( $ML$ ) is defined as

$$ML = \frac{k}{2} * \log_2 |S|,$$

with  $k$  the number of concepts in the partial hierarchy between the top concept and the concepts in the cut, and  $|S|$  the sample size, i.e., the total frequency of the data set. The *data length* ( $DL$ ) is defined as

$$DL = - \sum_{n \in S} \log_2 p(n).$$

---

<sup>3</sup>A *cut* is defined as a set of concepts in the concept hierarchy that defines a partition of the "leaf" concepts (the lowest concepts in the hierarchy), viewing each concept in the cut as representing the set of all leaf concepts it dominates.

The probability of a noun  $p(n)$  is determined by dividing the total probability of the concept class the noun belongs to,  $p(\text{concept})$ , by the size of that class,  $|\text{concept}|$ , i.e., the number of nouns that are dominated by that concept:

$$p(n) = \frac{p(\text{concept})}{|\text{concept}|}.$$

The higher the concept within the hierarchy, the more nouns receive an equal probability, and the greater is the data length.

The probability of the concept class in turn is determined by dividing the frequency of the concept class  $f(\text{concept})$  by the sample size:

$$p(\text{concept}) = \frac{f(\text{concept})}{|S|},$$

where  $f(\text{concept})$  is calculated by upward propagation of the frequencies of the nominal lexemes from the data sample through the hierarchy. For example, if the nouns *coffee*, *tea*, *milk* appeared with frequencies 25, 50, 3, respectively, within a specific argument slot, then their hypernym concept *beverage* would be assigned a frequency of 78, and these 78 would be propagated further upwards to the next hypernyms, etc. As a result, each concept class is assigned a fraction of the frequency of the whole data set (and the top concept receives the total frequency of the data set). For calculating  $p(\text{concept})$  (and the overall data length), though, only the concept classes within the cut through the hierarchy are relevant.

Our model uses WordNet 3.0 as the concept hierarchy, and comprises one (complete) a priori WordNet model for the lexical head probabilities  $p(a|r)$  and one (partial) model for each selectional probability distribution  $p(r|c, f, i)$ , cf. Section 2.1.

## 2.4 Combining EM and MDL

The training procedure that combines the EM training with the MDL principle can be summarised as follows.

1. The probabilities of a verb class model with  $c$  classes and a pre-defined set of verbs and frames are initialised randomly. The selectional preference models start out with the most general WordNet concept only, i.e., the partial WordNet hierarchies underlying the probabilities  $p(r|c, f, i)$  initially only contain the concept  $r$  for *entity*.

2. The model is trained for a pre-defined number of iterations. In each iteration, not only the model probabilities are re-estimated and maximised (as done by EM), but also the cuts through the concept hierarchies that represent the various selectional preference models are re-assessed. In each iteration, the following steps are performed.

(a) The partial WordNet hierarchies that represent the selectional preference models are expanded to include the hyponyms of the respective leaf concepts of the partial hierarchies. I.e., in the first iteration, all models are expanded towards the hyponyms of *entity*, and in subsequent iterations each selectional preference model is expanded to include the hyponyms of the leaf nodes in the partial hierarchies resulting from the previous iteration. This expansion step allows the selection models to become more and more detailed, as the training proceeds and the verb clusters (and their selectional restrictions) become increasingly specific.

(b) The training tuples are processed: For each tuple, a PCFG parse forest as indicated by Figure 1 is done, and the Inside-Outside algorithm is applied to estimate the frequencies of the "parse tree rules", given the current model probabilities.

(c) The MDL principle is applied to each selectional preference model: Starting from the respective leaf concepts in the partial hierarchies, MDL is calculated to compare each set of hyponym concepts that share a hypernym with the respective hypernym concept. If the MDL is lower for the set of hyponyms than the hypernym, the hyponyms are left in the partial hierarchy. Otherwise the expansion of the hypernym towards the hyponyms is undone and we continue recursively upwards the hierarchy, calculating MDL to compare the former hypernym and its co-hyponyms with the next upper hypernym, etc. The recursion allows the training algorithm to remove nodes which were added in earlier iterations and are no longer relevant. It stops if the MDL is lower for the hyponyms than for the hypernym.

This step results in selectional preference models that minimally contain the top concept *entity*, and maximally contain the partial WordNet hierarchy between *entity* and the concept classes that have been expanded within this iteration.

(d) The probabilities of the verb class model are

maximised based on the frequency estimates obtained in step (b).

### 3 Experiments

The model is generally applicable to all languages for which WordNet exists, and for which the WordNet functions provided by Princeton University are available. For the purposes of this paper, we choose English as a case study.

#### 3.1 Experimental Setup

The input data for training the verb class models were derived from Viterbi parses of the whole British National Corpus, using the lexicalised PCFG for English by Carroll and Rooth (1998). We took only active clauses into account, and disregarded auxiliary and modal verbs as well as particle verbs, leaving a total of 4,852,371 Viterbi parses. Those input tuples were then divided into 90% training data and 10% test data, providing 4,367,130 training tuples (over 2,769,804 types), and 485,241 test tuples (over 368,103 types).

As we wanted to train and assess our verb class model under various conditions, we used different fractions of the training data in different training regimes. Because of time and memory constraints, we only used training tuples that appeared at least twice. (For the sake of comparison, we also trained one model on all tuples.) Furthermore, we disregarded tuples with personal pronoun arguments; they are not represented in WordNet, and even if they are added (e.g. to general concepts such as *person*, *entity*) they have a rather destructive effect. We considered two subsets of the subcategorisation frames with 10 and 20 elements, which were chosen according to their overall frequency in the training data; for example, the 10 most frequent frame types were *subj:obj*, *subj*, *subj:ap*, *subj:to*, *subj:obj:obj2*, *subj:obj:pp-in*, *subj:adv*, *subj:pp-in*, *subj:vbase*, *subj:that*.<sup>4</sup> When relying on these 10/20 subcategorisation frames, plus including the above restrictions, we were left with 39,773/158,134 and 42,826/166,303 training tuple types/tokens, respectively. The overall number of training tuples

<sup>4</sup>A frame lists its arguments, separated by ':'. Most arguments within the frame types should be self-explanatory. *ap* is an adjectival phrase.



was therefore much smaller than the generally available data. The corresponding numbers including tuples with a frequency of one were 478,717/597,078 and 577,755/701,232.

The number of clusters in the experiments was either 20 or 50, and we used up to 50 iterations over the training tuples. The model probabilities were output after each 5th iteration. The output comprises all model probabilities introduced in Section 2.1. The following sections describe the evaluation of the experiments, and the results.

### 3.2 Evaluation

One of the goals in the development of the presented verb class model was to obtain an accurate statistical model of verb-argument tuples, i.e. a model which precisely predicts the tuple probabilities. In order to evaluate the performance of the model in this respect, we conducted an evaluation experiment, in which we computed the probability which the verb class model assigns to our test tuples and compared it to the corresponding probability assigned by a baseline model. The model with the higher probability is judged the better model.

We expected that the verb class model would perform better than the baseline model on tuples where one or more of the arguments were not observed with the respective verb, because either the argument itself or a semantically similar argument (according to the selectional preferences) was observed with verbs belonging to the same cluster. We also expected that the verb class model assigns a lower probability than the baseline model to test tuples which frequently occurred in the training data, since the verb class model fails to describe precisely the idiosyncratic properties of verbs which are not shared by the other verbs of its cluster.

**The Baseline Model** The baseline model decomposes the probability of a verb-argument tuple into a product of conditional probabilities:<sup>5</sup>

$$p(v, f, a_1^{n_f}) = p(v) p(f|v) \prod_{i=1}^{n_f} p(a_i | a_1^{i-1}, \langle v, f \rangle, f_i)$$

<sup>5</sup>  $f_i$  is the label of the  $i^{\text{th}}$  slot. The verb and the subcategorisation frame are enclosed in angle brackets because they are treated as a unit during smoothing.

The probability of our example tuple  $\langle \text{*speak*, subj-pp.to, professor, audience} \rangle$  in the baseline model is then  $p(\text{*speak*}) p(\text{subj-pp.to}|\text{*speak*}) p(\text{professor}|\langle \text{*speak*, subj-pp.to} \rangle, \text{subj}) p(\text{audience}|\text{professor}, \langle \text{*speak*, subj-pp.to} \rangle, \text{pp.to})$ .

The model contains no hidden variables. Thus the parameters can be directly estimated from the training data with relative frequencies. The parameter estimates are smoothed with modified Kneser-Ney smoothing (Chen and Goodman, 1998), such that the probability of each tuple is positive.

**Smoothing of the Verb Class Model** Although the verb class model has a built-in smoothing capacity, it needs additional smoothing for two reasons: Firstly, some of the nouns in the test data did not occur in the training data. The verb class model assigns a zero probability to such nouns. Hence we smoothed the concept instantiation probabilities  $p(\text{noun}|\text{concept})$  with Witten-Bell smoothing (Chen and Goodman, 1998). Secondly, we smoothed the probabilities of the concepts in the selectional preference models where zero probabilities may occur.

The smoothing ensures that the verb class model assigns a positive probability to each verb-argument tuple with a known verb, a known subcategorisation frame, and arguments which are in WordNet. Other tuples were excluded from the evaluation because the verb class model cannot deal with them.

### 3.3 Results

The evaluation results of our classification experiments are presented in Table 1, for 20 and 50 clusters, with 10 and 20 subcategorisation frame types. The table cells provide the  $\log_e$  of the probabilities per tuple token. The probabilities increase with the number of iterations, flattening out after approx. 25 iterations, as illustrated by Figure 2. Both for 10 and 20 frames, the results are better for 50 than for 20 clusters, with small differences between 10 and 20 frames. The results vary between -11.850 and -10.620 (for 5-50 iterations), in comparison to baseline values of -11.546 and -11.770 for 10 and 20 frames, respectively. The results thus show that our verb class model results are above the baseline results after 10 iterations; this means that our statistical model then assigns higher probabilities to the test tuples than the baseline model.

No. of Clusters	Iteration									
	5	10	15	20	25	30	35	40	45	50
<i>10 frames</i>										
20	-11.770	-11.408	-10.978	-10.900	-10.853	-10.841	-10.831	-10.823	-10.817	-10.812
50	-11.850	-11.452	-11.061	-10.904	-10.730	-10.690	-10.668	-10.628	-10.625	-10.620
<i>20 frames</i>										
20	-11.769	-11.430	-11.186	-10.971	-10.921	-10.899	-10.886	-10.875	-10.873	-10.869
50	-11.841	-11.472	-11.018	-10.850	-10.737	-10.728	-10.706	-10.680	-10.662	-10.648

Table 1: Clustering results – BNC tuples.

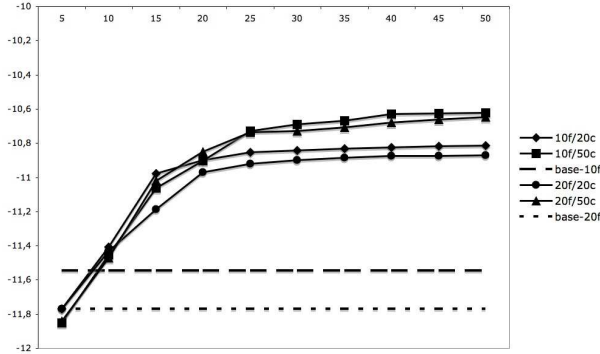


Figure 2: Illustration of clustering results.

Including input tuples with a frequency of one in the training data with 10 subcategorisation frames (as mentioned in Section 3.1) decreases the  $\log_e$  per tuple to between -13.151 and -12.498 (for 5-50 iterations), with similar training behaviour as in Figure 2, and in comparison to a baseline of -17.988. The differences in the result indicate that the models including the hapax legomena are worse than the models that excluded the sparse events; at the same time, the differences between baseline and clustering model are larger.

In order to get an intuition about the qualitative results of the clusterings, we select two example clusters that illustrate that the idea of the verb class model has been realised within the clusters. According to our own intuition, the clusters are overall semantically impressive, beyond the examples. Future work will assess by semantics-based evaluations of the clusters (such as pseudo-word disambiguation, or a comparison against existing verb classifications), whether this intuition is justified, whether it transfers to the majority of verbs within the cluster analyses, and whether the clusters capture polysemic verbs appropriately.

The two examples are taken from the 10 frame/50 cluster verb class model, with probabilities of 0.05 and 0.04. The ten most probable verbs in the first cluster are *show*, *suggest*, *indicate*, *reveal*, *find*, *imply*, *conclude*, *demonstrate*, *state*, *mean*, with the two most probable frame types *subj* and *subj:that*, i.e., the intransitive frame, and a frame that subcategorises a *that* clause. As selectional preferences within the intransitive frame (and quite similarly in the *subj:that* frame), the most probable concept classes<sup>6</sup> are *study*, *report*, *survey*, *name*, *research*, *result*, *evidence*. The underlined nouns represent specific concept classes, because they are leaf nodes in the selectional preference hierarchy, thus referring to very specific selectional preferences, which are potentially useful for collocation induction. The ten most probable verbs in the second cluster are *arise*, *remain*, *exist*, *continue*, *need*, *occur*, *change*, *improve*, *begin*, *become*, with the intransitive frame being most probable. The most probable concept classes are *problem*, *condition*, *question*, *natural phenomenon*, *situation*. The two examples illustrate that the verbs within a cluster are semantically related, and that they share obvious subcategorisation frames with intuitively plausible selectional preferences.

## 4 Related Work

Our model is an extension of and thus most closely related to the latent semantic clustering (LSC) model (Rooth et al., 1999) for verb-argument pairs  $\langle v, a \rangle$  which defines their probability as follows:

$$p(v, a) = \sum_c p(c) p(v|c) p(a|c)$$

In comparison to our model, the LSC model only considers a single argument (such as direct objects),

<sup>6</sup>For readability, we only list one noun per WordNet concept.

or a fixed number of arguments from one particular subcategorisation frame, whereas our model defines a probability distribution over all subcategorisation frames. Furthermore, our model specifies selectional preferences in terms of general WordNet concepts rather than sets of individual words.

In a similar vein, our model is both similar and distinct in comparison to the soft clustering approaches by Pereira et al. (1993) and Korhonen et al. (2003). Pereira et al. (1993) suggested deterministic annealing to cluster verb-argument pairs into classes of verbs and nouns. On the one hand, their model is asymmetric, thus not giving the same interpretation power to verbs and arguments; on the other hand, the model provides a more fine-grained clustering for nouns, in the form of an additional hierarchical structure of the noun clusters. Korhonen et al. (2003) used verb-frame pairs (instead of verb-argument pairs) to cluster verbs relying on the Information Bottleneck (Tishby et al., 1999). They had a focus on the interpretation of verbal polysemy as represented by the soft clusters. The main difference of our model in comparison to the above two models is, again, that we incorporate selectional preferences (rather than individual words, or subcategorisation frames).

In addition to the above soft-clustering models, various approaches towards semantic verb classification have relied on hard-clustering models, thus simplifying the notion of verbal polysemy. Two large-scale approaches of this kind are Schulte im Walde (2006), who used k-Means on verb subcategorisation frames and verbal arguments to cluster verbs semantically, and Joanis et al. (2008), who applied Support Vector Machines to a variety of verb features, including subcategorisation slots, tense, voice, and an approximation to animacy. To the best of our knowledge, Schulte im Walde (2006) is the only hard-clustering approach that previously incorporated selectional preferences as verb features. However, her model was not soft-clustering, and she only used a simple approach to represent selectional preferences by WordNet's top-level concepts, instead of making use of the whole hierarchy and more sophisticated methods, as in the current paper.

Last but not least, there are other models of selectional preferences than the MDL model we used in our paper. Most such models also rely on the

WordNet hierarchy (Resnik, 1997; Abney and Light, 1999; Ciaramita and Johnson, 2000; Clark and Weir, 2002). Brockmann and Lapata (2003) compared some of the models against human judgements on the acceptability of sentences, and demonstrated that the models were significantly correlated with human ratings, and that no model performed best; rather, the different methods are suited for different argument relations.

## 5 Summary and Outlook

This paper presented an innovative, complex approach to semantic verb classes that relies on selectional preferences as verb properties. The probabilistic verb class model underlying the semantic classes was trained by a combination of the EM algorithm and the MDL principle, providing soft clusters with two dimensions (verb senses and subcategorisation frames with selectional preferences) as a result. A language model-based evaluation showed that after 10 training iterations the verb class model results are above the baseline results.

We plan to improve the verb class model with respect to (i) a concept-wise (instead of a cut-wise) implementation of the MDL principle, to operate on concepts instead of combinations of concepts; and (ii) variations of the concept hierarchy, using e.g. the sense-clustered WordNets from the Stanford WordNet Project (Snow et al., 2007), or a WordNet version improved by concepts from DOLCE (Gangemi et al., 2003), to check on the influence of conceptual details on the clustering results. Furthermore, we aim to use the verb class model in NLP tasks, (i) as resource for lexical induction of verb senses, verb alternations, and collocations, and (ii) as a lexical resource for the statistical disambiguation of parse trees.

## References

- Steven Abney and Marc Light. 1999. Hiding a Semantic Class Hierarchy in a Markov Model. In *Proceedings of the ACL Workshop on Unsupervised Learning in Natural Language Processing*, pages 1–8, College Park, MD.
- Leonard E. Baum. 1972. An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. *Inequalities*, III:1–8.

- Carsten Brockmann and Mirella Lapata. 2003. Evaluating and Combining Approaches to Selectional Preference Acquisition. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 27–34, Budapest, Hungary.
- Glenn Carroll and Mats Rooth. 1998. Valence Induction with a Head-Lexicalized PCFG. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing*, Granada, Spain.
- Stanley Chen and Joshua Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University.
- Massimiliano Ciaramita and Mark Johnson. 2000. Explaining away Ambiguity: Learning Verb Selectional Preference with Bayesian Networks. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 187–193, Saarbrücken, Germany.
- Stephen Clark and David Weir. 2002. Class-Based Probability Estimation using a Semantic Hierarchy. *Computational Linguistics*, 28(2):187–206.
- Bonnie J. Dorr and Doug Jones. 1996. Role of Word Sense Disambiguation in Lexical Acquisition: Predicting Semantics from Syntactic Cues. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 322–327, Copenhagen, Denmark.
- Aldo Gangemi, Nicola Guarino, Claudio Masolo, and Alessandro Oltramari. 2003. Sweetening WordNet with DOLCE. *AI Magazine*, 24(3):13–24.
- Eric Joanis, Suzanne Stevenson, and David James. 2008? A General Feature Space for Automatic Verb Classification. *Natural Language Engineering*. To appear.
- Judith L. Klavans and Min-Yen Kan. 1998. The Role of Verbs in Document Analysis. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, pages 680–686, Montreal, Canada.
- Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 868–876, Prague, Czech Republic.
- Upali S. Kohomban and Wee Sun Lee. 2005. Learning Semantic Classes for Word Sense Disambiguation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 34–41, Ann Arbor, MI.
- Anna Korhonen, Yuval Krymolowski, and Zvika Marx. 2003. Clustering Polysemic Subcategorization Frame Distributions Semantically. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 64–71, Sapporo, Japan.
- Anna Korhonen. 2002. *Subcategorization Acquisition*. Ph.D. thesis, University of Cambridge, Computer Laboratory. Technical Report UCAM-CL-TR-530.
- Karim Lari and Steve J. Young. 1990. The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4:35–56.
- Hang Li and Naoki Abe. 1998. Generalizing Case Frames Using a Thesaurus and the MDL Principle. *Computational Linguistics*, 24(2):217–244.
- Paola Merlo and Suzanne Stevenson. 2001. Automatic Verb Classification Based on Statistical Distributions of Argument Structure. *Computational Linguistics*, 27(3):373–408.
- Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional Clustering of English Words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, OH.
- Detlef Prescher, Stefan Riezler, and Mats Rooth. 2000. Using a Probabilistic Class-Based Lexicon for Lexical Ambiguity Resolution. In *Proceedings of the 18th International Conference on Computational Linguistics*.
- Philip Resnik. 1997. Selectional Preference and Sense Disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, Washington, DC.
- Jorma Rissanen. 1978. Modeling by Shortest Data Description. *Automatica*, 14:465–471.
- Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a Semantically Annotated Lexicon via EM-Based Clustering. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, Maryland, MD.
- Sabine Schulte im Walde. 2006. Experiments on the Automatic Induction of German Semantic Verb Classes. *Computational Linguistics*, 32(2):159–194.
- Eric V. Siegel and Kathleen R. McKeown. 2000. Learning Methods to Combine Linguistic Indicators: Improving Aspectual Classification and Revealing Linguistic Insights. *Computational Linguistics*, 26(4):595–628.
- Rion Snow, Sushant Prakash, Daniel Jurafsky, and Andrew Y. Ng. 2007. Learning to Merge Word Senses. In *Proceedings of the joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, Czech Republic.
- Naftali Tishby, Fernando Pereira, and William Bialek. 1999. The Information Bottleneck Method. In *Proceedings of the 37th Annual Conference on Communication, Control, and Computing*, Monticello, IL.

# Randomized Language Models via Perfect Hash Functions

David Talbot\*

School of Informatics  
University of Edinburgh  
2 Buccleuch Place, Edinburgh, UK  
d.r.talbot@sms.ed.ac.uk

Thorsten Brants

Google Inc.  
1600 Amphitheatre Parkway  
Mountain View, CA 94303, USA  
brants@google.com

## Abstract

We propose a succinct randomized language model which employs a *perfect hash function* to encode *fingerprints* of  $n$ -grams and their associated probabilities, backoff weights, or other parameters. The scheme can represent any standard  $n$ -gram model and is easily combined with existing model reduction techniques such as entropy-pruning. We demonstrate the space-savings of the scheme via machine translation experiments within a distributed language modeling framework.

## 1 Introduction

Language models (LMs) are a core component in statistical machine translation, speech recognition, optical character recognition and many other areas. They distinguish plausible word sequences from a set of candidates. LMs are usually implemented as  $n$ -gram models parameterized for each distinct sequence of up to  $n$  words observed in the training corpus. Using higher-order models and larger amounts of training data can significantly improve performance in applications, however the size of the resulting LM can become prohibitive.

With large monolingual corpora available in major languages, making use of all the available data is now a fundamental challenge in language modeling. Efficiency is paramount in applications such as machine translation which make huge numbers of LM requests per sentence. To scale LMs to larger corpora with higher-order dependencies, researchers

have considered alternative parameterizations such as class-based models (Brown et al., 1992), model reduction techniques such as entropy-based pruning (Stolcke, 1998), novel representation schemes such as suffix arrays (Emami et al., 2007), Golomb Coding (Church et al., 2007) and distributed language models that scale more readily (Brants et al., 2007).

In this paper we propose a novel randomized language model. Recent work (Talbot and Osborne, 2007b) has demonstrated that randomized encodings can be used to represent  $n$ -gram counts for LMs with significant space-savings, circumventing information-theoretic constraints on lossless data structures by allowing errors with some small probability. In contrast the representation scheme used by our model encodes parameters directly. It can be combined with any  $n$ -gram parameter estimation method and existing model reduction techniques such as entropy-based pruning. Parameters that are stored in the model are retrieved without error; however, *false positives* may occur whereby  $n$ -grams not in the model are incorrectly ‘found’ when requested. The false positive rate is determined by the space usage of the model.

Our randomized language model is based on the Bloomier filter (Chazelle et al., 2004). We encode *fingerprints* (random hashes) of  $n$ -grams together with their associated probabilities using a *perfect hash function* generated at random (Majewski et al., 1996). Lookup is very efficient: the values of 3 cells in a large array are combined with the fingerprint of an  $n$ -gram. This paper focuses on machine translation. However, many of our findings should transfer to other applications of language modeling.

---

\*Work completed while this author was at Google Inc.

## 2 Scaling Language Models

In statistical machine translation (SMT), LMs are used to score candidate translations in the target language. These are typically  $n$ -gram models that approximate the probability of a word sequence by assuming each token to be independent of all but  $n - 1$  preceding tokens. Parameters are estimated from monolingual corpora with parameters for each distinct word sequence of length  $l \in [n]$  observed in the corpus. Since the number of parameters grows somewhat exponentially with  $n$  and linearly with the size of the training corpus, the resulting models can be unwieldy even for relatively small corpora.

### 2.1 Scaling Strategies

Various strategies have been proposed to scale LMs to larger corpora and higher-order dependencies. *Model-based* techniques seek to parameterize the model more efficiently (e.g. latent variable models, neural networks) or to reduce the model size directly by pruning uninformative parameters, e.g. (Stolcke, 1998), (Goodman and Gao, 2000). *Representation-based* techniques attempt to reduce space requirements by representing the model more efficiently or in a form that scales more readily, e.g. (Emami et al., 2007), (Brants et al., 2007), (Church et al., 2007).

### 2.2 Lossy Randomized Encodings

A fundamental result in information theory (Carter et al., 1978) states that a random set of objects cannot be stored using constant space per object as the universe from which the objects are drawn grows in size: the space required to uniquely identify an object increases as the set of possible objects from which it must be distinguished grows. In language modeling the universe under consideration is the set of all possible  $n$ -grams of length  $n$  for given vocabulary. Although  $n$ -grams observed in natural language corpora are *not* randomly distributed within this universe no lossless data structure that we are aware of can circumvent this space-dependency on both the  $n$ -gram order and the vocabulary size. Hence as the training corpus and vocabulary grow, a model will require more space *per parameter*.

However, if we are willing to accept that occasionally our model will be unable to distinguish between distinct  $n$ -grams, then it is possible to store

each parameter in constant space independent of both  $n$  and the vocabulary size (Carter et al., 1978), (Talbot and Osborne, 2007a). The space required in such a *lossy encoding* depends only on the range of values associated with the  $n$ -grams and the desired error rate, i.e. the probability with which two distinct  $n$ -grams are assigned the same fingerprint.

### 2.3 Previous Randomized LMs

Recent work (Talbot and Osborne, 2007b) has used lossy encodings based on Bloom filters (Bloom, 1970) to represent logarithmically quantized corpus statistics for language modeling. While the approach results in significant space savings, working with corpus statistics, rather than  $n$ -gram probabilities directly, is computationally less efficient (particularly in a distributed setting) and introduces a dependency on the smoothing scheme used. It also makes it difficult to leverage existing model reduction strategies such as entropy-based pruning that are applied to final parameter estimates.

In the next section we describe our randomized LM scheme based on perfect hash functions. This scheme can be used to encode any standard  $n$ -gram model which may first be processed using any conventional model reduction technique.

## 3 Perfect Hash-based Language Models

Our randomized LM is based on the Bloomier filter (Chazelle et al., 2004). We assume the  $n$ -grams and their associated parameter values have been precomputed and stored on disk. We then encode the model in an array such that each  $n$ -gram's value can be retrieved. Storage for this array is the model's only significant space requirement once constructed.<sup>1</sup>

The model uses randomization to map  $n$ -grams to *fingerprints* and to generate a *perfect hash function* that associates  $n$ -grams with their values. The model can erroneously return a value for an  $n$ -gram that was never actually stored, but will always return the correct value for an  $n$ -gram that is in the model. We will describe the randomized algorithm used to encode  $n$ -gram parameters in the model, analyze the probability of a false positive, and explain how we construct and query the model in practice.

<sup>1</sup>Note that we do not store the  $n$ -grams explicitly and therefore that the model's parameter set cannot easily be enumerated.

### 3.1 $N$ -gram Fingerprints

We wish to encode a set of  $n$ -gram/value pairs

$$\mathcal{S} = \{(x_1, v(x_1)), (x_2, v(x_2)), \dots, (x_N, v(x_N))\}$$

using an array  $A$  of size  $M$  and a perfect hash function. Each  $n$ -gram  $x_i$  is drawn from some set of possible  $n$ -grams  $\mathcal{U}$  and its associated value  $v(x_i)$  from a corresponding set of possible values  $\mathcal{V}$ .

We do not store the  $n$ -grams and their probabilities directly but rather encode a *fingerprint* of each  $n$ -gram  $f(x_i)$  together with its associated value  $v(x_i)$  in such a way that the value can be retrieved when the model is queried with the  $n$ -gram  $x_i$ .

A fingerprint hash function  $f : \mathcal{U} \rightarrow [0, B - 1]$  maps  $n$ -grams to integers between 0 and  $B - 1$ .<sup>2</sup> The array  $A$  in which we encode  $n$ -gram/value pairs has addresses of size  $\lceil \log_2 B \rceil$  hence  $B$  will determine the amount of space used per  $n$ -gram. There is a trade-off between space and error rate since the larger  $B$  is, the lower the probability of a false positive. This is analyzed in detail below. For now we assume only that  $B$  is at least as large as the range of values stored in the model, i.e.  $B \geq |\mathcal{V}|$ .

### 3.2 Composite Perfect Hash Functions

The function used to associate  $n$ -grams with their values (Eq. (1)) combines a composite perfect hash function (Majewski et al., 1996) with the fingerprint function. An example is shown in Fig. 1. The composite hash function is made up of  $k$  independent hash functions  $h_1, h_2, \dots, h_k$  where each  $h_i : \mathcal{U} \rightarrow [0, M - 1]$  maps  $n$ -grams to locations in the array  $A$ . The lookup function is then defined as  $g : \mathcal{U} \rightarrow [0, B - 1]$  by<sup>3</sup>

$$g(x_i) = f(x_i) \otimes \left( \bigotimes_{i=1}^k A[h_i(x_i)] \right) \quad (1)$$

where  $f(x_i)$  is the fingerprint of  $n$ -gram  $x_i$  and  $A[h_i(x_i)]$  is the value stored in location  $h_i(x_i)$  of the array  $A$ . Eq. (1) is evaluated to retrieve an  $n$ -gram's parameter during decoding. To encode our model correctly we must ensure that  $g(x_i) = v(x_i)$  for all  $n$ -grams in our set  $\mathcal{S}$ . Generating  $A$  to encode this

<sup>2</sup>The analysis assumes that all hash functions are random.

<sup>3</sup>We use  $\otimes$  to denote the exclusive bitwise OR operator.

Composite hash function  $g(x)$  encodes  $v(x) = 3$

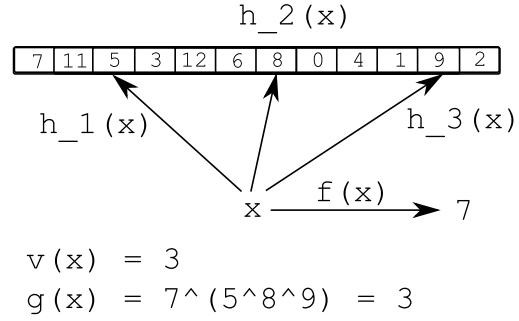


Figure 1: Encoding an  $n$ -gram's value in the array.

function for a given set of  $n$ -grams is a significant challenge described in the following sections.

### 3.3 Encoding $n$ -grams in the model

All addresses in  $A$  are initialized to zero. The procedure we use to ensure  $g(x_i) = v(x_i)$  for all  $x_i \in \mathcal{S}$  updates a single, unique location in  $A$  for each  $n$ -gram  $x_i$ . This location is chosen from among the  $k$  locations given by  $h_j(x_i), j \in [k]$ . Since the composite function  $g(x_i)$  depends on the values stored at all  $k$  locations  $A[h_1(x_i)], A[h_2(x_i)], \dots, A[h_k(x_i)]$  in  $A$ , we must also ensure that once an  $n$ -gram  $x_i$  has been encoded in the model, these  $k$  locations are not subsequently changed since this would invalidate the encoding; however,  $n$ -grams encoded later may reference earlier entries and therefore locations in  $A$  can effectively be 'shared' among parameters.

In the following section we describe a randomized algorithm to find a suitable order in which to enter  $n$ -grams in the model and, for each  $n$ -gram  $x_i$ , determine which of the  $k$  hash functions, say  $h_j$ , can be used to update  $A$  without invalidating previous entries. Given this ordering of the  $n$ -grams and the choice of hash function  $h_j$  for each  $x_i \in \mathcal{S}$ , it is clear that the following update rule will encode  $x_i$  in the array  $A$  so that  $g(x_i)$  will return  $v(x_i)$  (cf. Eq.(1))

$$A[h_j(x_i)] = v(x_i) \otimes f(x_i) \otimes \bigotimes_{i=1, i \neq j}^k A[h_i(x_i)]. \quad (2)$$

### 3.4 Finding an Ordered Matching

We now describe an algorithm (Algorithm 1; (Majewski et al., 1996)) that selects one of the  $k$  hash

functions  $h_j, j \in [k]$  for each  $n$ -gram  $x_i \in \mathcal{S}$  and an order in which to apply the update rule Eq. (2) so that  $g(x_i)$  maps  $x_i$  to  $v(x_i)$  for all  $n$ -grams in  $\mathcal{S}$ .

This problem is equivalent to finding an *ordered matching* in a bipartite graph whose LHS nodes correspond to  $n$ -grams in  $\mathcal{S}$  and RHS nodes correspond to locations in  $A$ . The graph initially contains edges from each  $n$ -gram to each of the  $k$  locations in  $A$  given by  $h_1(x_i), h_2(x_i), \dots, h_k(x_i)$  (see Fig. (2)). The algorithm uses the fact that any RHS node that has degree one (i.e. a single edge) can be safely matched with its associated LHS node since no remaining LHS nodes can be dependent on it.

We first create the graph using the  $k$  hash functions  $h_j, j \in [k]$  and store a list (`degree_one`) of those RHS nodes (locations) with degree one. The algorithm proceeds by removing nodes from `degree_one` in turn, pairing each RHS node with the unique LHS node to which it is connected. We then remove both nodes from the graph and push the pair  $(x_i, h_j(x_i))$  onto a stack (`matched`). We also remove any other edges from the matched LHS node and add any RHS nodes that now have degree one to `degree_one`. The algorithm succeeds if, while there are still  $n$ -grams left to match, `degree_one` is never empty. We then encode  $n$ -grams in the order given by the stack (i.e., first-in-last-out).

Since we remove each location in  $A$  (RHS node) from the graph as it is matched to an  $n$ -gram (LHS node), each location will be associated with at most one  $n$ -gram for updating. Moreover, since we match an  $n$ -gram to a location only once the location has degree one, we are guaranteed that any other  $n$ -grams that depend on this location are already on the stack and will therefore only be encoded once we have updated this location. Hence dependencies in  $g$  are respected and  $g(x_i) = v(x_i)$  will remain true following the update in Eq. (2) for each  $x_i \in \mathcal{S}$ .

### 3.5 Choosing Random Hash Functions

The algorithm described above is not guaranteed to succeed. Its success depends on the size of the array  $M$ , the number of  $n$ -grams stored  $|\mathcal{S}|$  and the choice of random hash functions  $h_j, j \in [k]$ . Clearly we require  $M \geq |\mathcal{S}|$ ; in fact, an argument from Majewski et al. (1996) implies that if  $M \geq 1.23|\mathcal{S}|$  and  $k = 3$ , the algorithm succeeds with high probabil-

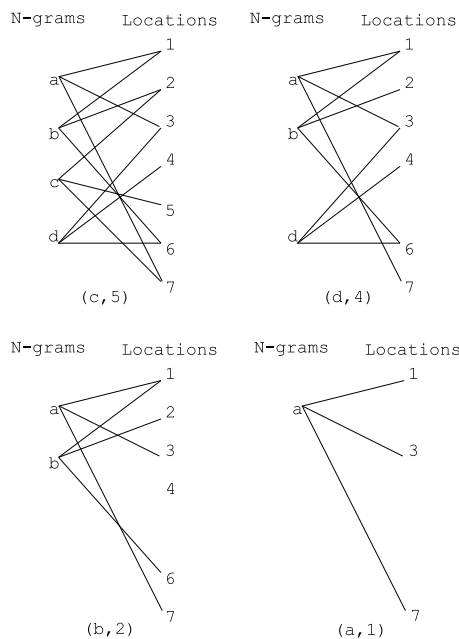


Figure 2: The ordered matching algorithm: `matched` =  $[(a, 1), (b, 2), (d, 4), (c, 5)]$

ity. We use 2-universal hash functions (L. Carter and M. Wegman, 1979) defined for a range of size  $M$  via a prime  $P \geq M$  and two random numbers  $1 \leq a_j \leq P$  and  $0 \leq b_j \leq P$  for  $j \in [k]$  as

$$h_j(x) \equiv a_j x + b_j \pmod{P}$$

taken modulo  $M$ . We generate a set of  $k$  hash functions by sampling  $k$  pairs of random numbers  $(a_j, b_j), j \in [k]$ . If the algorithm does not find a matching with the current set of hash functions, we re-sample these parameters and re-start the algorithm. Since the probability of failure on a single attempt is low when  $M \geq 1.23|\mathcal{S}|$ , the probability of failing multiple times is very small.

### 3.6 Querying the Model and False Positives

The construction we have described above ensures that for any  $n$ -gram  $x_i \in \mathcal{S}$  we have  $g(x_i) = v(x_i)$ , i.e., we retrieve the correct value. To retrieve a value given an  $n$ -gram  $x_i$  we simply compute the fingerprint  $f(x_i)$ , the hash functions  $h_j(x_i), j \in [k]$  and then return  $g(x_i)$  using Eq. (1). Note that unlike the constructions in (Talbot and Osborne, 2007b) and (Church et al., 2007) no errors are possible for  $n$ -grams stored in the model. Hence we will not make errors for common  $n$ -grams that are typically in  $\mathcal{S}$ .



---

**Algorithm 1** Ordered Matching

---

**Input** : Set of  $n$ -grams  $\mathcal{S}$ ;  $k$  hash functions  $h_j, j \in [k]$ ; number of available locations  $M$ .  
**Output** : Ordered matching `matched` or `FAIL`.  
`matched`  $\leftarrow []$   
**for all**  $i \in [0, M - 1]$  **do**  
   $r2l_i \leftarrow \emptyset$   
**end for**  
**for all**  $x_i \in \mathcal{S}$  **do**  
   $l2r_i \leftarrow \emptyset$   
  **for all**  $j \in [k]$  **do**  
     $l2r_i \leftarrow l2r_i \cup h_j(x_i)$   
     $r2l_{h_j(x_i)} \leftarrow r2l_{h_j(x_i)} \cup x_i$   
  **end for**  
**end for**  
`degree_one`  $\leftarrow \{i \in [0, M - 1] \mid |r2l_i| = 1\}$   
**while**  $|\text{degree\_one}| \geq 1$  **do**  
  `rhs`  $\leftarrow \text{POP } \text{degree\_one}$   
  `lhs`  $\leftarrow \text{POP } r2l_{\text{rhs}}$   
  PUSH (`lhs`, `rhs`) onto `matched`  
  **for all**  $rhs' \in l2r_{\text{lhs}}$  **do**  
    POP  $r2l_{\text{rhs}'}$   
    **if**  $|r2l_{\text{rhs}'}| = 1$  **then**  
      `degree_one`  $\leftarrow \text{degree\_one} \cup \text{rhs}'$   
    **end if**  
  **end for**  
**end while**  
**if**  $|\text{matched}| = |\mathcal{S}|$  **then**  
  **return** `matched`  
**else**  
  **return** `FAIL`  
**end if**

---

On the other hand, querying the model with an  $n$ -gram that was not stored, i.e. with  $x_i \in \mathcal{U} \setminus \mathcal{S}$  we may erroneously return a value  $v \in \mathcal{V}$ .

Since the fingerprint  $f(x_i)$  is assumed to be distributed uniformly at random (u.a.r.) in  $[0, B - 1]$ ,  $g(x_i)$  is also u.a.r. in  $[0, B - 1]$  for  $x_i \in \mathcal{U} \setminus \mathcal{S}$ . Hence with  $|\mathcal{V}|$  values stored in the model, the probability that  $x_i \in \mathcal{U} \setminus \mathcal{S}$  is assigned a value in  $v \in \mathcal{V}$  is

$$\Pr\{g(x_i) \in \mathcal{V} \mid x_i \in \mathcal{U} \setminus \mathcal{S}\} = |\mathcal{V}|/B.$$

We refer to this event as a *false positive*. If  $\mathcal{V}$  is fixed, we can obtain a false positive rate  $\epsilon$  by setting  $B$  as

$$B \equiv |\mathcal{V}|/\epsilon.$$

For example, if  $|\mathcal{V}|$  is 128 then taking  $B = 1024$  gives an error rate of  $\epsilon = 128/1024 = 0.125$  with each entry in  $A$  using  $\lceil \log_2 1024 \rceil = 10$  bits. Clearly  $B$  must be at least  $|\mathcal{V}|$  in order to distinguish each value. We refer to the additional bits allocated to

each location (i.e.  $\lceil \log_2 B \rceil - \log_2 |\mathcal{V}|$  or 3 in our example) as *error bits* in our experiments below.

### 3.7 Constructing the Full Model

When encoding a large set of  $n$ -gram/value pairs  $\mathcal{S}$ , Algorithm 1 will only be practical if the raw data and graph can be held in memory as the perfect hash function is generated. This makes it difficult to encode an extremely large set  $\mathcal{S}$  into a single array  $A$ . The solution we adopt is to split  $\mathcal{S}$  into  $t$  smaller sets  $\mathcal{S}'_i, i \in [t]$  that are arranged in lexicographic order.<sup>4</sup> We can then encode each subset in a separate array  $A'_i, i \in [t]$  in turn in memory. Querying each of these arrays for each  $n$ -gram requested would be inefficient and inflate the error rate since a false positive could occur on each individual array. Instead we store an index of the final  $n$ -gram encoded in each array and given a request for an  $n$ -gram's value, perform a binary search for the appropriate array.

### 3.8 Sanity Checks

Our models are consistent in the following sense

$$(w_1, w_2, \dots, w_n) \in \mathcal{S} \implies (w_2, \dots, w_n) \in \mathcal{S}.$$

Hence we can infer that an  $n$ -gram can *not* be present in the model, if the  $n - 1$ -gram consisting of the final  $n - 1$  words has already tested false. Following (Talbot and Osborne, 2007a) we can avoid unnecessary false positives by not querying for the longer  $n$ -gram in such cases.

Backoff smoothing algorithms typically request the longest  $n$ -gram supported by the model first, requesting shorter  $n$ -grams only if this is not found. In our case, however, if a query is issued for the 5-gram  $(w_1, w_2, w_3, w_4, w_5)$  when only the unigram  $(w_5)$  is present in the model, the probability of a false positive using such a backoff procedure would not be  $\epsilon$  as stated above, but rather the probability that we fail to avoid an error on *any* of the four queries performed prior to requesting the unigram, i.e.  $1 - (1 - \epsilon)^4 \approx 4\epsilon$ . We therefore query the model first with the unigram working up to the full  $n$ -gram requested by the decoder only if the preceding queries test positive. The probability of returning a false positive for any  $n$ -gram requested by the decoder (but not in the model) will then be at most  $\epsilon$ .

<sup>4</sup>In our system we use subsets of 5 million  $n$ -grams which can easily be encoded using less than 2GB of working space.

## 4 Experimental Set-up

### 4.1 Distributed LM Framework

We deploy the randomized LM in a distributed framework which allows it to scale more easily by distributing it across multiple language model servers. We encode the model stored on each language model server using the randomized scheme.

The proposed randomized LM can encode parameters estimated using any smoothing scheme (e.g. Kneser-Ney, Katz etc.). Here we choose to work with *stupid backoff* smoothing (Brants et al., 2007) since this is significantly more efficient to train and deploy in a distributed framework than a *context-dependent* smoothing scheme such as Kneser-Ney. Previous work (Brants et al., 2007) has shown it to be appropriate to large-scale language modeling.

### 4.2 LM Data Sets

The language model is trained on four data sets:

**target:** The English side of Arabic-English parallel data provided by LDC (132 million tokens).

**gigaword:** The English Gigaword dataset provided by LDC (3.7 billion tokens).

**webnews:** Data collected over several years, up to January 2006 (34 billion tokens).

**web:** The Web 1T 5-gram Version 1 corpus provided by LDC (1 trillion tokens).<sup>5</sup>

An initial experiment will use the Web 1T 5-gram corpus only; all other experiments will use a log-linear combination of models trained on each corpus. The combined model is pre-compiled with weights trained on development data by our system.

### 4.3 Machine Translation

The SMT system used is based on the framework proposed in (Och and Ney, 2004) where translation is treated as the following optimization problem

$$\hat{\mathbf{e}} = \arg \max_{\mathbf{e}} \sum_{i=1}^M \lambda_i \Phi_i(\mathbf{e}, \mathbf{f}). \quad (3)$$

Here  $\mathbf{f}$  is the source sentence that we wish to translate,  $\mathbf{e}$  is a translation in the target language,  $\Phi_i, i \in [M]$  are feature functions and  $\lambda_i, i \in [M]$  are weights. (Some features may not depend on  $\mathbf{f}$ .)

<sup>5</sup> $N$ -grams with count  $< 40$  are not included in this data set.

	Full Set	Entropy-Pruned
# 1-grams	13,588,391	13,588,391
# 2-grams	314,843,401	184,541,402
# 3-grams	977,069,902	439,430,328
# 4-grams	1,313,818,354	407,613,274
# 5-grams	1,176,470,663	238,348,867
Total	3,795,790,711	1,283,522,262

Table 1: Num. of  $n$ -grams in the Web 1T 5-gram corpus.

## 5 Experiments

This section describes three sets of experiments: first, we encode the Web 1T 5-gram corpus as a randomized language model and compare the resulting size with other representations; then we measure false positive rates when requesting  $n$ -grams for a held-out data set; finally we compare translation quality when using conventional (lossless) languages models and our randomized language model.

Note that the standard practice of measuring perplexity is not meaningful here since (1) for efficient computation, the language model is not normalized; and (2) even if this were not the case, quantization and false positives would render it unnormalized.

### 5.1 Encoding the Web 1T 5-gram corpus

We build a language model from the Web 1T 5-gram corpus. Parameters, corresponding to negative logarithms of relative frequencies, are quantized to 8-bits using a uniform quantizer. More sophisticated quantizers (e.g. (S. Lloyd, 1982)) may yield better results but are beyond the scope of this paper.

Table 1 provides some statistics about the corpus. We first encode the full set of  $n$ -grams, and then a version that is reduced to approx. 1/3 of its original size using entropy pruning (Stolcke, 1998).

Table 2 shows the total space and number of bytes required per  $n$ -gram to encode the model under different schemes: “LDC gzip’d” is the size of the files as delivered by LDC; “Trie” uses a compact trie representation (e.g., (Clarkson et al., 1997; Church et al., 2007)) with 3 byte word ids, 1 byte values, and 3 byte indices; “Block encoding” is the encoding used in (Brants et al., 2007); and “randomized” uses our novel randomized scheme with 12 error bits. The latter requires around 60% of the space of the next best representation and less than half of the com-

	size (GB)	bytes/ $n$ -gram
<b>Full Set</b>		
LDC gzip'd	24.68	6.98
Trie	21.46	6.07
Block Encoding	18.00	5.14
Randomized	10.87	3.08
<b>Entropy Pruned</b>		
Trie	7.70	6.44
Block Encoding	6.20	5.08
Randomized	3.68	3.08

Table 2: Web 1T 5-gram language model sizes with different encodings. “Randomized” uses 12 error bits.

monly used trie encoding. Our method is the only one to use the same amount of space per parameter for both full and entropy-pruned models.

## 5.2 False Positive Rates

All  $n$ -grams explicitly inserted into our randomized language model are retrieved without error; however,  $n$ -grams not stored may be incorrectly assigned a value resulting in a false positive. Section (3) analyzed the theoretical error rate; here, we measure error rates in practice when retrieving  $n$ -grams for approx. 11 million tokens of previously unseen text (news articles published after the training data had been collected). We measure this separately for all  $n$ -grams of order 2 to 5 from the same text.

The language model is trained on the four data sources listed above and contains 24 billion  $n$ -grams. With 8-bit parameter values, the model requires 55.2/69.0/82.7 GB storage when using 8/12/16 error bits respectively (this corresponds to 2.46/3.08/3.69 bytes/ $n$ -gram).

Using such a large language model results in a large fraction of known  $n$ -grams in new text. Table 3 shows, e.g., that almost half of all 5-grams from the new text were seen in the training data.

Column (1) in Table 4 shows the number of false positives that occurred for this test data. Column (2) shows this as a fraction of the number of unseen  $n$ -grams in the data. This number should be close to  $2^{-b}$  where  $b$  is the number of error bits (i.e. 0.003906 for 8 bits and 0.000244 for 12 bits). The error rates for bigrams are close to their expected values. The numbers are much lower for higher  $n$ -gram orders due to the use of sanity checks (see Section 3.8).

	total	seen	unseen
2gms	11,093,093	98.98%	1.02%
3gms	10,652,693	91.08%	8.92%
4gms	10,212,293	68.39%	31.61%
5gms	9,781,777	45.51%	54.49%

Table 3: Number of  $n$ -grams in test set and percentages of  $n$ -grams that were seen/unseen in the training data.

	(1) false pos.	(2) $\frac{\text{false pos}}{\text{unseen}}$	(3) $\frac{\text{false pos}}{\text{total}}$
<b>8 error bits</b>			
2gms	376	0.003339	0.000034
3gms	2839	0.002988	0.000267
4gms	6659	0.002063	0.000652
5gms	6356	0.001192	0.000650
total	16230	0.001687	0.000388
<b>12 error bits</b>			
2gms	25	0.000222	0.000002
3gms	182	0.000192	0.000017
4gms	416	0.000129	0.000041
5gms	407	0.000076	0.000042
total	1030	0.000107	0.000025

Table 4: False positive rates with 8 and 12 error bits.

The overall fraction of  $n$ -grams requested for which an error occurs is of most interest in applications. This is shown in Column (3) and is around a factor of 4 smaller than the values in Column (2). On average, we expect to see 1 error in around 2,500 requests when using 8 error bits, and 1 error in 40,000 requests with 12 error bits (see “total” row).

## 5.3 Machine Translation

We run an improved version of our 2006 NIST MT Evaluation entry for the Arabic-English “Unlimited” data track.<sup>6</sup> The language model is the same one as in the previous section.

Table 5 shows baseline translation BLEU scores for a lossless (non-randomized) language model with parameter values quantized into 5 to 8 bits. We use MT04 data for system development, with MT05 data and MT06 (“NIST” subset) data for blind testing. As expected, results improve when using more bits. There seems to be little benefit in going beyond

<sup>6</sup>See <http://www.nist.gov/speech/tests/mt/2006/doc/>

bits	dev	test	test
	MT04	MT05	MT06
5	0.5237	0.5608	0.4636
6	0.5280	0.5671	0.4649
7	0.5299	0.5691	0.4672
8	0.5304	0.5697	0.4663

Table 5: Baseline BLEU scores with lossless  $n$ -gram model and different quantization levels (bits).

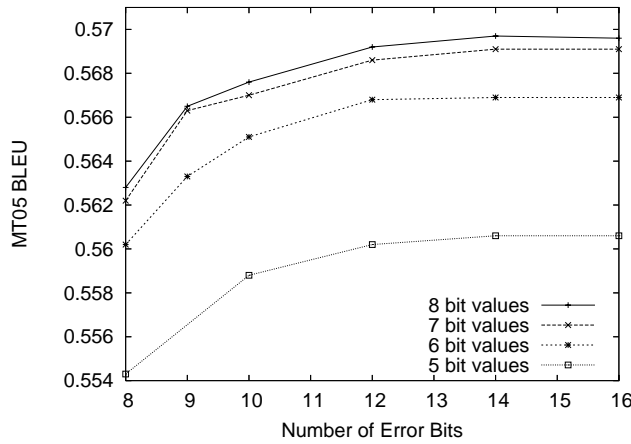


Figure 3: BLEU scores on the MT05 data set.

8 bits. Overall, our baseline results compare favorably to those reported on the NIST MT06 web site.

We now replace the language model with a randomized version. Fig. 3 shows BLEU scores for the MT05 evaluation set with parameter values quantized into 5 to 8 bits and 8 to 16 additional ‘error’ bits. Figure 4 shows a similar graph for MT06 data. We again see improvements as quantization uses more bits. There is a large drop in performance when reducing the number of error bits from 10 to 8, while increasing it beyond 12 bits offers almost no further gains with scores that are almost identical to the lossless model. Using 8-bit quantization and 12 error bits results in an overall requirement of  $(8 + 12) \times 1.23 = 24.6$  bits = 3.08 bytes per  $n$ -gram.

All runs use the sanity checks described in Section 3.8. Without sanity checks, scores drop, e.g. by 0.002 for 8-bit quantization and 12 error bits.

Randomization and entropy pruning can be combined to achieve further space savings with minimal loss in quality as shown in Table (6). The BLEU score drops by between 0.0007 to 0.0018 while the

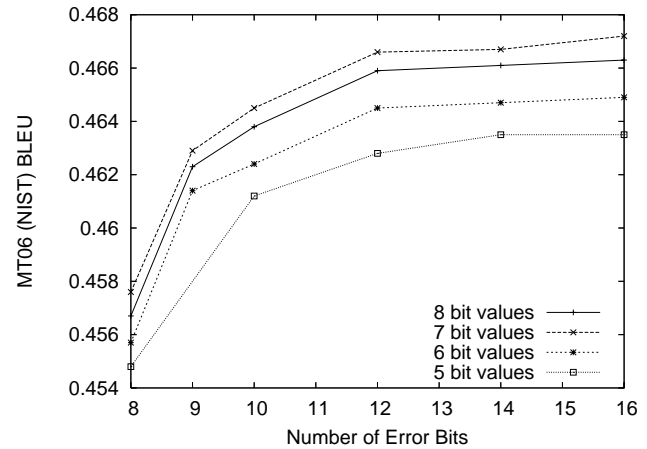


Figure 4: BLEU scores on MT06 data (“NIST” subset).

LM	size	dev	test	test
	GB	MT04	MT05	MT06
unpruned block	116	0.5304	0.5697	0.4663
unpruned rand	69	0.5299	0.5692	0.4659
pruned block	42	0.5294	0.5683	0.4665
pruned rand	27	0.5289	0.5679	0.4656

Table 6: Combining randomization and entropy pruning. All models use 8-bit values; “rand” uses 12 error bits.

model is reduced to approx. 1/4 of its original size.

## 6 Conclusions

We have presented a novel randomized language model based on perfect hashing. It can associate arbitrary parameter types with  $n$ -grams. Values explicitly inserted into the model are retrieved without error; false positives may occur but are controlled by the number of bits used per  $n$ -gram. The amount of storage needed is independent of the size of the vocabulary and the  $n$ -gram order. Lookup is very efficient: the values of 3 cells in a large array are combined with the fingerprint of an  $n$ -gram.

Experiments have shown that this randomized language model can be combined with entropy pruning to achieve further memory reductions; that error rates occurring in practice are much lower than those predicted by theoretical analysis due to the use of runtime sanity checks; and that the same translation quality as a lossless language model representation can be achieved when using 12 ‘error’ bits, resulting in approx. 3 bytes per  $n$ -gram (this includes one byte to store parameter values).

## References

- B. Bloom. 1970. Space/time tradeoffs in hash coding with allowable errors. *CACM*, 13:422–426.
- Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-CoNLL 2007*, Prague.
- Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.
- Peter Brown, Stephen Della Pietra, Vincent Della Pietra, and Robert Mercer. 1993. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.
- Larry Carter, Robert W. Floyd, John Gill, George Markowsky, and Mark N. Wegman. 1978. Exact and approximate membership testers. In *STOC*, pages 59–65.
- L. Carter and M. Wegman. 1979. Universal classes of hash functions. *Journal of Computer and System Science*, 18:143–154.
- Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. 2004. The Bloomier Filter: an efficient data structure for static support lookup tables. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 30–39.
- Kenneth Church, Ted Hart, and Jianfeng Gao. 2007. Compressing trigram language models with golomb coding. In *Proceedings of EMNLP-CoNLL 2007*, Prague, Czech Republic, June.
- P. Clarkson and R. Rosenfeld. 1997. Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings of EUROSPEECH*, vol. 1, pages 2707–2710, Rhodes, Greece.
- Ahmad Emami, Kishore Papineni, and Jeffrey Sorensen. 2007. Large-scale distributed language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2007*, Hawaii, USA.
- J. Goodman and J. Gao. 2000. Language model size reduction by pruning and clustering. In *ICSLP'00*, Beijing, China.
- S. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.
- B.S. Majewski, N.C. Wormald, G. Havas, and Z.J. Czech. 1996. A family of perfect hashing methods. *British Computer Journal*, 39(6):547–554.
- Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.
- Andreas Stolcke. 1998. Entropy-based pruning of back-off language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.
- D. Talbot and M. Osborne. 2007a. Randomised language modelling for statistical machine translation. In *45th Annual Meeting of the ACL 2007, Prague*.
- D. Talbot and M. Osborne. 2007b. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *EMNLP/CoNLL 2007, Prague*.

# Applying Morphology Generation Models to Machine Translation

**Kristina Toutanova**  
Microsoft Research  
Redmond, WA, USA

**Hisami Suzuki**  
Microsoft Research  
Redmond, WA, USA

**Achim Ruopp**  
Butler Hill Group  
Redmond, WA, USA

kristout@microsoft.com hisamis@microsoft.com v-acruop@microsoft.com

## Abstract

We improve the quality of statistical machine translation (SMT) by applying models that predict word forms from their stems using extensive morphological and syntactic information from both the source and target languages. Our inflection generation models are trained independently of the SMT system. We investigate different ways of combining the inflection prediction component with the SMT system by training the base MT system on fully inflected forms or on word stems. We applied our inflection generation models in translating English into two morphologically complex languages, Russian and Arabic, and show that our model improves the quality of SMT over both phrasal and syntax-based SMT systems according to BLEU and human judgments.

## 1 Introduction

One of the outstanding problems for further improving machine translation (MT) systems is the difficulty of dividing the MT problem into sub-problems and tackling each sub-problem in isolation to improve the overall quality of MT. Evidence for this difficulty is the fact that there has been very little work investigating the use of such independent sub-components, though we started to see some successful cases in the literature, for example in word alignment (Fraser and Marcu, 2007), target language capitalization (Wang et al., 2006) and case marker generation (Toutanova and Suzuki, 2007).

This paper describes a successful attempt to integrate a subcomponent for generating word inflections into a statistical machine translation (SMT)

system. Our research is built on previous work in the area of using morpho-syntactic information for improving SMT. Work in this area is motivated by two advantages offered by morphological analysis: (1) it provides linguistically motivated clustering of words and makes the data less sparse; (2) it captures morphological constraints applicable on the target side, such as agreement phenomena. This second problem is very difficult to address with word-based translation systems, when the relevant morphological information in the target language is either non-existent or implicitly encoded in the source language. These two aspects of morphological processing have often been addressed separately: for example, morphological pre-processing of the input data is a common method of addressing the first aspect, e.g. (Goldwater and McClosky, 2005), while the application of a target language model has almost solely been responsible for addressing the second aspect. Minkov et al. (2007) introduced a way to address these problems by using a rich feature-based model, but did not apply the model to MT.

In this paper, we integrate a model that predicts target word inflection in the translations of English into two morphologically complex languages (Russian and Arabic) and show improvements in the MT output. We study several alternative methods for integration and show that it is best to propagate uncertainty among the different components as shown by other research, e.g. (Finkel et al., 2006), and in some cases, to factor the translation problem so that the baseline MT system can take advantage of the reduction in sparsity by being able to work on word stems. We also demonstrate that our independently trained models are portable, showing that they can improve both syntactic and phrasal SMT systems.

## 2 Related work

There has been active research on incorporating morphological knowledge in SMT. Several approaches use pre-processing schemes, including segmentation of clitics (Lee, 2004; Habash and Sadat, 2006), compound splitting (Nießen and Ney, 2004) and stemming (Goldwater and McClosky, 2005). Of these, the segmentation approach is difficult to apply when the target language is morphologically rich as the segmented morphemes must be put together in the output (El-Kahlout and Oflazer, 2006); and in fact, most work using pre-processing focused on translation into English. In recent work, Koehn and Hoang (2007) proposed a general framework for including morphological features in a phrase-based SMT system by factoring the representation of words into a vector of morphological features and allowing a phrase-based MT system to work on any of the factored representations, which is implemented in the Moses system. Though our motivation is similar to that of Koehn and Hoang (2007), we chose to build an independent component for inflection prediction in isolation rather than folding morphological information into the main translation model. While this may lead to search errors due to the fact that the models are not integrated as tightly as possible, it offers some important advantages, due to the very decoupling of the components. First, our approach is not affected by restrictions on the allowable context size or a phrasal segmentation that are imposed by current MT decoders. This also makes the model portable and applicable to different types of MT systems. Second, we avoid the problem of the combinatorial expansion in the search space which currently arises in the factored approach of Moses.

Our inflection prediction model is based on (Minkov et al., 2007), who build models to predict the inflected forms of words in Russian and Arabic, but do not apply their work to MT. In contrast, we focus on methods of integration of an inflection prediction model with an MT system, and on evaluation of the model’s impact on translation. Other work closely related to ours is (Toutanova and Suzuki, 2007), which uses an independently trained case marker prediction model in an English-Japanese translation system, but it focuses on the problem of generating a small set of closed class words rather

than generating inflected forms for each word in translation, and proposes different methods of integration of the components.

## 3 Inflection prediction models

This section describes the task and our model for inflection prediction, following (Minkov et al., 2007).

We define the task of inflection prediction as the task of choosing the correct inflections of given target language stems, given a corresponding source sentence. The stemming and inflection operations we use are defined by lexicons.

### 3.1 Lexicon operations

For each target language we use a lexicon  $L$  which determines the following necessary operations:

*Stemming*: returns the set of possible morphological stems  $S_w = \{s^1, \dots, s^l\}$  for the word  $w$  according to  $L$ .<sup>1</sup>

*Inflection*: returns the set of surface word forms  $I_w = \{i^1, \dots, i^m\}$  for the stems  $S_w$  according to  $L$ .

*Morphological analysis*: returns the set of possible morphological analyses  $A_w = \{a^1, \dots, a^v\}$  for  $w$ . A morphological analysis  $a$  is a vector of categorical values, where each dimension and its possible values are defined by  $L$ .

For the morphological analysis operation, we used the same set of morphological features described in (Minkov et al., 2007), that is, seven features for Russian (POS, Person, Number, Gender, Tense, Mood and Case) and 12 for Arabic (POS, Person, Number, Gender, Tense, Mood, Negation, Determiner, Conjunction, Preposition, Object and Possessive pronouns). Each word is factored into a stem (uninflected form) and a subset of these features, where features can have either binary (as in Determiner in Arabic) or multiple values. Some features are relevant only for a particular (set of) part-of-speech (POS) (e.g., Gender is relevant only in nouns, pronouns, verbs, and adjectives in Russian), while others combine with practically all categories (e.g., Conjunction in Arabic). The number of possible inflected forms per stem is therefore quite large: as we see in Table 1 of Section 3, there are on average 14 word forms per stem in Russian and 24 in

<sup>1</sup>Alternatively, stemming can return a disambiguated stem analysis; in which case the set  $S_w$  consists of one item. The same is true with the operation of morphological analysis.

Arabic for our dataset. This makes the generation of correct forms a challenging problem in MT.

The Russian lexicon was obtained by intersecting a general domain lexicon with our training data (Table 2), and the Arabic lexicon was obtained by running the Buckwalter morphological analyser (Buckwalter, 2004) on the training data. Contextual disambiguation of morphology was not performed in either of these languages. In addition to the forms supposed by our lexicon, we also treated capitalization as an inflectional feature in Russian, and defined all true-case word variants as possible inflections of its stem(s). Arabic does not use capitalization.

### 3.2 Task

More formally, our task is as follows: given a source sentence  $e$ , a sequence of stems in the target language  $S_1, \dots, S_t, \dots, S_n$  forming a translation of  $e$ , and additional morpho-syntactic annotations  $A$  derived from the input, select an inflection  $y_t$  from its inflection set  $I_t$  for every stem set  $S_t$  in the target sentence.

### 3.3 Models

We built a Maximum Entropy Markov model for inflection prediction following (Minkov et al., 2007). The model decomposes the probability of an inflection sequence into a product of local probabilities for the prediction for each word. The local probabilities are conditioned on the previous  $k$  predictions ( $k$  is set to four in Russian and two in Arabic in our experiments). The probability of a predicted inflection sequence, therefore, is given by:

$$p(\bar{y} | \bar{x}) = \prod_{t=1}^n p(y_t | y_{t-1} \dots y_{t-k}, x_t), y_t \in I_t,$$

where  $I_t$  is the set of inflections corresponding to  $S_t$ , and  $x_t$  refers to the *context* at position  $t$ . The context available to the task includes extensive morphological and syntactic information obtained from the aligned source and target sentences. Figure 1 shows an example of an aligned English-Russian sentence pair: on the source (English) side, POS tags and word dependency structure are indicated by solid arcs. The alignments between English and Russian words are indicated by the dotted lines. The dependency structure on the Russian side, indicated by solid arcs, is given by a treelet MT system (see Section 4.1), projected from the word dependency struc-

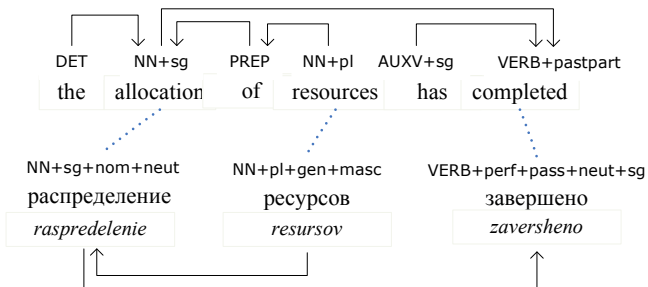


Figure 1: Aligned English-Russian sentence pair with syntactic and morphological annotation.

ture of English and word alignment information.

The features for our inflection prediction model are binary and pair up predicates on the *context* ( $\bar{x}, y_{t-1} \dots y_{t-k}$ ) and the *target label* ( $y_t$ ). The features at a certain position  $t$  can refer to any word in the source sentence, any word stem in the target language, or any morpho-syntactic information in  $A$ . This is the source of the power of a model used as an independent component – because it does not need to be integrated in the main search of an MT decoder, it is not subject to the decoder’s locality constraints, and can thus make use of more global information.

### 3.4 Performance on reference translations

Table 1 summarizes the results of applying the inflection prediction model on *reference* translations, simulating the ideal case where the translations input to our model contain correct stems in correct order. We stemmed the reference translations, predicted the inflection for each stem, and measured the accuracy of prediction, using a set of sentences that were not part of the training data (1K sentences were used for Arabic and 5K for Russian).<sup>2</sup> Our model performs significantly better than both the random and trigram language model baselines, and achieves an accuracy of over 91%, which suggests that the model is effective when its input is clean in its stem choice and order. Next, we apply our model in the more noisy but realistic scenario of predicting inflections of MT output sentences.

<sup>2</sup>The accuracy is based on the words in our lexicon. We define the stem of an out-of-vocabulary (OOV) word to be itself, so in the MT scenario described below, we will not predict the word forms for an OOV item, and will simply leave it unchanged.



	Russian	Arabic
Random	16.4	8.7
LM	81.0	69.4
Model	91.6	91.0
Avg   <i>I</i>	13.9	24.1

Table 1: Results on reference translations (accuracy, %).

## 4 Machine translation systems and data

We integrated the inflection prediction model with two types of machine translation systems: systems that make use of syntax and surface phrase-based systems.

### 4.1 Treelet translation system

This is a syntactically-informed MT system, designed following (Quirk et al., 2005). In this approach, translation is guided by treelet translation pairs, where a treelet is a connected subgraph of a syntactic dependency tree. Translations are scored according to a linear combination of feature functions. The features are similar to the ones used in phrasal systems, and their weights are trained using max-BLEU training (Och, 2003). There are nine feature functions in the treelet system, including log-probabilities according to inverted and direct channel models estimated by relative frequency, lexical weighting channel models following Vogel et al. (2003), a trigram target language model, two order models, word count, phrase count, and average phrase size functions.

The treelet translation model is estimated using a parallel corpus. First, the corpus is word-aligned using an implementation of lexicalized-HMMs (He, 2007); then the source sentences are parsed into a dependency structure, and the dependency is projected onto the target side following the heuristics described in (Quirk et al., 2005). These aligned sentence pairs form the training data of the inflection models as well. An example was given in Figure 1.

### 4.2 Phrasal translation system

This is a re-implementation of the Pharaoh translation system (Koehn, 2004). It uses the same lexicalized-HMM model for word alignment as the treelet system, and uses the standard extraction heuristics to extract phrase pairs using forward and backward alignments. In decoding, the system uses a linear combination of feature functions whose

weights are trained using max-BLEU training. The features include log-probabilities according to inverted and direct channel models estimated by relative frequency, lexical weighting channel models, a trigram target language model, distortion, word count and phrase count.

### 4.3 Data sets

For our English-Russian and English-Arabic experiments, we used data from a technical (computer) domain. For each language pair, we used a set of parallel sentences (**train**) for training the MT system sub-models (e.g., phrase tables, language model), a set of parallel sentences (**lambda**) for training the combination weights with max-BLEU training, a set of parallel sentences (**dev**) for training a small number of combination parameters for our integration methods (see Section 5), and a set of parallel sentences (**test**) for final evaluation. The details of these sets are shown in Table 2. The training data for the inflection models is always a subset of the training set (**train**). All MT systems for a given language pair used the same datasets.

Dataset	sent pairs	word tokens (avg/sent)	
<b>English-Russian</b>			
		English	Russian
train	1,642K	24,351K (14.8)	22,002K (13.4)
lambda	2K	30K (15.1)	27K (13.7)
dev	1K	14K (13.9)	13K (13.5)
test	4K	61K (15.3)	60K (14.9)
<b>English-Arabic</b>			
		English	Arabic
train	463K	5,223K (11.3)	4,761K (10.3)
lambda	2K	22K (11.1)	20K (10.0)
dev	1K	11K (11.1)	10K (10.0)
test	4K	44K (11.0)	40K (10.1)

Table 2: Data set sizes, rounded up to the nearest 1000.

## 5 Integration of inflection models with MT systems

We describe three main methods of integration we have considered. The methods differ in the extent to which the factoring of the problem into two subproblems — predicting stems and predicting inflections — is reflected in the base MT systems. In the first method, the MT system is trained to produce fully inflected target words and the inflection model can change the inflections. In the other two methods, the

MT system is trained to produce sequences of target language stems  $\mathbf{S}$ , which are then inflected by the inflection component. Before we motivate these methods, we first describe the general framework for integrating our inflection model into the MT system.

For each of these methods, we assume that the output of the base MT system can be viewed as a ranked list of translation hypotheses for each source sentence  $e$ . More specifically, we assume an output  $\{\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_m\}$  of  $m$ -best translations which are sequences of target language stems. The translations further have scores  $\{w_1, w_2, \dots, w_m\}$  assigned by the base MT system. We also assume that each translation hypothesis  $\mathbf{S}_i$  together with source sentence  $e$  can be annotated with the annotation  $A$ , as illustrated in Figure 1. We discuss how we convert the output of the base MT systems to this form in the subsections below.

Given such a list of candidate stem sequences, the base MT model together with the inflection model and a language model choose a translation  $\mathbf{Y}^*$  as follows:

$$(1) Y_i = \arg \max_{Y'_i \in \text{Inf}(S_i)} \lambda_1 \log P_{IM}(Y'_i | \mathbf{S}_i) + \lambda_2 \log P_{LM}(Y'_i), i = 1 \dots n$$

$$(2) Y^* = \arg \max_{i=1 \dots n} \lambda_1 \log P_{IM}(Y_i | \mathbf{S}_i) + \lambda_2 \log P_{LM}(Y_i) + \lambda_3 w_i$$

In these formulas, the dependency on  $e$  and  $A$  is omitted for brevity in the expression for the probability according to the inflection model  $P_{IM}$ .  $P_{LM}(Y'_i)$  is the joint probability of the sequence of inflected words according to a trigram language model (LM). The LM used for the integration is the same LM used in the base MT system that is trained on fully inflected word forms (the base MT system trained on stems uses an LM trained on a stem sequence). Equation (1) shows that the model first selects the best sequence of inflected forms for each MT hypothesis  $\mathbf{S}_i$  according to the LM and the inflection model. Equation (2) shows that from these  $n$  fully inflected hypotheses, the model then selects the one which has the best score, combined with the base MT score  $w_i$  for  $S_i$ . We should note that this method does not represent standard  $n$ -best re-ranking because the input from the base MT system contains sequences of stems, and the model is generating fully inflected translations from them. Thus the chosen translation may not be in the provided  $n$ -best list. This method is more similar to the one used

in (Wang et al., 2006), with the difference that they use only 1-best input from a base MT system.

The interpolation weights  $\lambda$  in Equations (1) and (2) as well as the optimal number of translations  $n$  from the base MT system to consider, given a maximum of  $m=100$  hypotheses, are trained using a separate dataset. We performed a grid search on the values of  $\lambda$  and  $n$ , to maximize the BLEU score of the final system on a development set (dev) of 1000 sentences (Table 2).

The three methods of integration differ in the way the base MT engine is applied. Since we always discard the choices of specific inflected forms for the target stems by converting candidate translations to sequences of stems, it is interesting to know whether we need a base MT system that produces fully inflected translations or whether we can do as well or better by training the base MT systems to produce sequences of stems. Stemming the target sentences is expected to be helpful for word alignment, especially when the stemming operation is defined so that the word alignment becomes more one-to-one (Goldwater and McClosky, 2005). In addition, stemming the target sentences reduces the sparsity in the translation tables and language model, and is likely to impact positively the performance of an MT system in terms of its ability to recover correct sequences of stems in the target. Also, machine learning tells us that solving a more complex problem than we are evaluated on (in our case for the base MT, predicting stems together with their inflections instead of just predicting stems) is theoretically unjustified (Vapnik, 1995).

However, for some language pairs, stemming one language can make word alignment worse, if it leads to more violations in the assumptions of current word alignment models, rather than making the source look more like the target. In addition, using a trigram LM on stems may lead to larger violations of the Markov independence assumptions, than using a trigram LM on fully inflected words. Thus, if we apply the exact same base MT system to use stemmed forms in alignment and/or translation, it is not a priori clear whether we would get a better result than if we apply the system to use fully inflected forms.

## 5.1 Method 1

In this method, the base MT system is trained in the usual way, from aligned pairs of source sentences and fully inflected target sentences. The inflection model is then applied to re-inflect the 1-best or  $m$ -best translations and to select an output translation. The hypotheses in the  $m$ -best output from the base MT system are stemmed and the scores of the stemmed hypotheses are assumed to be equal to the scores of the original ones.<sup>3</sup> Thus we obtain input of the needed form, consisting of  $m$  sequences of target language stems along with scores.

For this and other methods, if we are working with an  $m$ -best list from the treelet system, every translation hypothesis contains the annotations  $A$  that our model needs, because the system maintains the alignment, parse trees, etc., as part of its search space. Thus we do not need to do anything further to obtain input of the form necessary for application of the inflection model.

For the phrase-based system, we generated the annotations needed by first parsing the source sentence  $e$ , aligning the source and candidate translations with the word-alignment model used in training, and projected the dependency tree to the target using the algorithm of (Quirk et al., 2005). Note that it may be better to use the word alignment maintained as part of the translation hypotheses during search, but our solution is more suitable to situations where these can not be easily obtained.

For all methods, we study two settings for integration. In the first, we only consider ( $n=1$ ) hypotheses from the base MT system. In the second setting, we allow the model to use up to 100 translations, and to automatically select the best number to use. As seen in Table 3, ( $n=16$ ) translations were chosen for Russian and as seen in Table 5, ( $n=2$ ) were chosen for Arabic for this method.

## 5.2 Method 2

In this method, the base MT system is trained to produce sequences of stems in the target language. The most straightforward way to achieve this is to stem the training parallel data and to train the MT system using this input. This is our Method 3 described

<sup>3</sup>It may be better to take the max of the scores for a stem sequence occurring more than once in the list, or take the log-sum-exp of the scores.

below. We formulated Method 2 as an intermediate step, to decouple the impact of stemming at the alignment and translation stages.

In Method 2, word alignment is performed using fully inflected target language sentences. After alignment, the target language is stemmed and the base MT systems' sub-models are trained using this stemmed input and alignment. In addition to this word-aligned corpus the MT systems use another product of word alignment: the IBM model 1 translation tables. Because the trained translation tables of IBM model 1 use fully inflected target words, we generated stemmed versions of the translation tables by applying the rules of probability.

## 5.3 Method 3

In this method the base MT system produces sequences of target stems. It is trained in the same way as the baseline MT system, except its input parallel training data are preprocessed to stem the target sentences. In this method, stemming can impact word alignment in addition to the translation models.

## 6 MT performance results

Before delving into the results for each method, we discuss our evaluation measures. For automatically measuring performance, we used 4-gram BLEU against a single reference translation. We also report oracle BLEU scores which incorporate two kinds of oracle knowledge. For the methods using  $n=1$  translation from a base MT system, the oracle BLEU score is the BLEU score of the stemmed translation compared to the stemmed reference, which represents the upper bound achievable by changing only the inflected forms (but not stems) of the words in a translation. For models using  $n > 1$  input hypotheses, the oracle also measures the gain from choosing the best possible stem sequence in the provided ( $m=100$ -best) hypothesis list, in addition to choosing the best possible inflected forms for these stems. For the models in the tables, even if, say,  $n=16$  was chosen in parameter fitting, the oracle is measured on the initially provided list of 100-best.

### 6.1 English-Russian treelet system

Table 3 shows the results of the baseline and the model using the different methods for the treelet MT system on English-Russian. The baseline is the

Model	BLEU	Oracle BLEU
Base MT ( $n=1$ )	29.24	-
Method 1 ( $n=1$ )	30.44	36.59
Method 1 ( $n=16$ )	30.61	45.33
Method 2 ( $n=1$ )	30.79	37.38
Method 2 ( $n=16$ )	31.24	48.48
Method 3 ( $n=1$ )	31.42	38.06
Method 3 ( $n=32$ )	31.80	49.19

Table 3: Test set performance for English-to-Russian MT (BLEU) results by model using a treelet MT system.

treelet system described in Section 4.1 and trained on the data in Table 2.

We can see that Method 1 results in a good improvement of 1.2 BLEU points, even when using only the best ( $n = 1$ ) translation from the baseline. The oracle improvement achievable by predicting inflections is quite substantial: more than 7 BLEU points. Propagating the uncertainty of the baseline system by using more input hypotheses consistently improves performance across the different methods, with an additional improvement of between .2 and .4 BLEU points.

From the results of Method 2 we can see that reducing sparsity at translation modeling is advantageous. Both the oracle BLEU of the first hypothesis and the achieved performance of the model improved; the best performance achieved by Method 2 is .63 points higher than the performance of Method 1. We should note that the oracle performance for Method 2,  $n > 1$  is measured using 100-best lists of target stem sequences, whereas the one for Method 1 is measured using 100-best lists of inflected target words. This can be a disadvantage for Method 1, because a 100-best list of inflected translations actually contains about 50 different sequences of stems (the rest are distinctions in inflections). Nevertheless, even if we measure the oracle for Method 2 using 40-best, it is higher than the 100-best oracle of Method 1. In addition, it appears that using a hypothesis list larger than  $n > 1=100$  is not be helpful for our method, as the model chose to use only up to 32 hypotheses.

Finally, we can see that using stemming at the word alignment stage further improved both the oracle and the achieved results. The performance of the best model is 2.56 BLEU points better than the baseline. Since stemming in Russian for the most part removes properties of words which are not ex-

pressed in English at the word level, these results are consistent with previous results using stemming to improve word alignment. From these results, we also see that about half of the gain from using stemming in the base MT system came from improving word alignment, and half came from using translation models operating at the less sparse stem level.

Overall, the improvement achieved by predicting morphological properties of Russian words with a feature-rich component model is substantial, given the relatively large size of the training data (1.6 million sentences), and indicates that these kinds of methods are effective in addressing the problems in translating morphology-poor to morphology-rich languages.

## 6.2 English-Russian phrasal system

For the phrasal system, we performed integration only with Method 1, using the top 1 or 100-best translations. This is the most straightforward method for combining with any system, and we applied it as a proof-of-concept experiment.

Model	BLEU	Oracle BLEU
Base MT ( $n=1$ )	36.00	-
Method 1 ( $n=1$ )	36.43	42.33
Method 1 ( $n=100$ )	36.72	55.00

Table 4: Test set performance for English-to-Russian MT (BLEU) results by model using a phrasal MT system.

The phrasal MT system is trained on the same data as the treelet system. The phrase size and distortion limit were optimized (we used phrase size of 7 and distortion limit of 3). This system achieves a substantially better BLEU score (by 6.76) than the treelet system. The oracle BLEU score achievable by Method 1 using  $n=1$  translation, though, is still 6.3 BLEU point higher than the achieved BLEU.

Our model achieved smaller improvements for the phrasal system (0.43 improvement for  $n=1$  translations and 0.72 for the selected  $n=100$  translations). However, this improvement is encouraging given the large size of the training data. One direction for potentially improving these results is to use word alignments from the MT system, rather than using an alignment model to predict them.

Model	BLEU	Oracle BLEU
Base MT ( $n=1$ )	35.54	-
Method 1 ( $n=1$ )	37.24	42.29
Method 1 ( $n=2$ )	37.41	52.21
Method 2 ( $n=1$ )	36.53	42.46
Method 2 ( $n=4$ )	36.72	54.74
Method 3 ( $n=1$ )	36.87	42.96
Method 3 ( $n=2$ )	36.92	54.90

Table 5: Test set performance for English-to-Arabic MT (BLEU) results by model using a treelet MT system.

### 6.3 English-Arabic treelet system

The Arabic system also improves with the use of our mode: the best system (Method 1,  $n=2$ ) achieves the BLEU score of 37.41, a 1.87 point improvement over the baseline. Unlike the case of Russian, Method 2 and 3 do not achieve better results than Method 1, though the oracle BLEU score improves in these models (54.74 and 54.90 as opposed to 52.21 of Method 1). We do notice, however, that the oracle improvement for the 1-best analysis is much smaller than what we obtained in Russian.

We have been unable to closely diagnose why performance did not improve using Methods 2 and 3 so far due to the absence of expertise in Arabic, but one factor we suspect is affecting performance the most in Arabic is the definition of stemming: the effect of stemming is most beneficial when it is applied specifically to normalize the distinctions not explicitly encoded in the other language; it may hurt performance otherwise. We believe that in the case of Arabic, this latter situation is actually happening: grammatical properties explicitly encoded in English (e.g., definiteness, conjunction, pronominal clitics) are lost when the Arabic words are stemmed. This may be having a detrimental effect on the MT systems that are based on stemmed input. Further investigation is necessary to confirm this hypothesis.

### 6.4 Human evaluation

In this section we briefly report the results of human evaluation on the output of our inflection prediction system, as the correlation between BLEU scores and human evaluation results is not always obvious. We compared the output of our component against the best output of the treelet system without our component. We evaluated the following three scenarios: (1) Arabic Method 1 with  $n=1$ , which corresponds to the best performing system in BLEU according to

Table 5; (2) Russian, Method 1 with  $n=1$ ; (3) Russian, Method 3 with  $n=32$ , which corresponds to the best performing system in BLEU in Table 3. Note that in (1) and (2), the only differences in the compared outputs are the changes in word inflections, while in (3) the outputs may differ in the selection of the stems.

In all scenarios, two human judges (native speakers of these languages) evaluated 100 sentences that had different translations by the baseline system and our model. The judges were given the reference translations but not the source sentences, and were asked to classify each sentence pair into three categories: (1) the baseline system is better (score=-1), (2) the output of our model is better (score=1), or (3) they are of the same quality (score=0).

	human eval score	BLEU diff
Arabic Method 1	0.1	1.9
Russian Method 1	0.255	1.2
Russian Method 3	0.26	2.6

Table 6: Human evaluation results

Table 6 shows the results of the averaged, aggregated score across two judges per evaluation scenario, along with the BLEU score improvements achieved by applying our model. We see that in all cases, the human evaluation scores are positive, indicating that our models produce translations that are better than those produced by the baseline system.<sup>4</sup> We also note that in Russian, the human evaluation scores are similar for Method 1 and 3 (0.255 and 0.26), though the BLEU score gains are quite different (1.2 vs 2.6). This may be attributed to the fact that human evaluation typically favors the scenario where only word inflections are different (Toutanova and Suzuki, 2007).

## 7 Conclusion and future work

We have shown that an independent model of morphology generation can be successfully integrated with an SMT system, making improvements in both phrasal and syntax-based MT. In the future, we would like to include more sophistication in the design of a lexicon for a particular language pair based on error analysis, and extend our pre-processing to include other operations such as word segmentation.

<sup>4</sup>However, the improvement in Arabic is not statistically significant on this 100 sentence set.

## References

- Tim Buckwalter. 2004. Buckwalter arabic morphological analyzer version 2.0.
- Ilknur Durgar El-Kahlout and Kemal Oflazer. 2006. Initial explorations in English to Turkish statistical machine translation. In *NAACL workshop on statistical machine translation*.
- Jenny Finkel, Christopher Manning, and Andrew Ng. 2006. Solving the problem of cascading errors: approximate Bayesian inference for linguistic annotation pipelines. In *EMNLP*.
- Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303.
- Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *EMNLP*.
- Nizar Habash and Fatiha Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *HLT-NAACL*.
- Xiaodong He. 2007. Using word-dependent transition models in HMM based word alignment for statistical machine translation. In *ACL Workshop on Statistical Machine Translation*.
- Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *EMNLP-CoNLL*.
- Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *AMTA*.
- Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *HLT-NAACL*.
- Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *ACL*.
- Sonja Nießen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2):181–204.
- Franz Och. 2003. Minimum error rate training for statistical machine translation. In *ACL*.
- Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency tree translation: Syntactically informed phrasal SMT. In *ACL*.
- Kristina Toutanova and Hisami Suzuki. 2007. Generating case markers in machine translation. In *NAACL-HLT*.
- Vladimir Vapnik. 1995. *The nature of Statistical Learning Theory*. Springer-Verlag.
- Wei Wang, Kevin Knight, and Daniel Marcu. 2006. Capitalizing machine translation. In *HLT-NAACL*.

# Multilingual Harvesting of Cross-Cultural Stereotypes

**Tony Veale**

School of Computer Science  
University College Dublin  
Belfield, Dublin 4, Ireland  
tony.veale@ucd.ie

**Yanfen Hao**

School of Computer Science  
University College Dublin  
Belfield, Dublin 4, Ireland  
yanfen.hao@ucd.ie

**Guofu Li**

School of Computer Science  
University College Dublin  
Belfield, Dublin 4, Ireland  
li.guofu.l@gmail.com

## Abstract

People rarely articulate explicitly what a native speaker of a language is already assumed to know. So to acquire the stereotypical knowledge that underpins much of what is said in a given culture, one must look to what is implied by language rather than what is overtly stated. Similes are a convenient vehicle for this kind of knowledge, insofar as they mark out the most salient aspects of the most frequently evoked concepts. In this paper we perform a multilingual exploration of the space of common-place similes, by mining a large body of Chinese similes from the web and comparing these to the English similes harvested by Veale and Hao (2007). We demonstrate that while the simile-frame is inherently leaky in both languages, a multilingual analysis allows us to filter much of the noise that otherwise hinders the knowledge extraction process. In doing so, we can also identify a core set of stereotypical descriptions that exist in both languages and accurately map these descriptions onto a multilingual lexical ontology like HowNet. Finally, we demonstrate that conceptual descriptions that are derived from common-place similes are extremely compact and predictive of ontological structure.

## 1 Introduction

Direct perception of our environment is just one of the ways we can acquire knowledge of the world. Another, more distinctly human approach, is through the comprehension of linguistic descriptions of another person's perceptions and beliefs.

Since computers have limited means of human-like perception, the latter approach is also very much suited to the automatic acquisition of world knowledge by a computer (see Hearst, 1992; Charniak and Berland, 1999; Etzioni *et al.*, 2004; Völker *et al.*, 2005; Almuhareb and Poesio, 2005; Cimiano and Wenderoth, 2007; Veale and Hao, 2007). Thus, by using the web as a distributed text corpus (see Keller *et al.*, 2002), a multitude of facts and beliefs can be extracted, for purposes ranging from question-answering to ontology population.

The possible configurations of different concepts can also be learned from how the words denoting these concepts are distributed; thus, a computer can learn that coffee is a beverage that can be served hot or cold, white or black, strong or weak and sweet or bitter (see Almuhareb and Poesio, 2005). But it is difficult to discern from these facts the idealized or stereotypical states of the world, e.g., that one *expects* coffee to be hot and beer to be cold, so that if one spills coffee, we naturally infer the possibilities of scalding and staining without having to be told that the coffee was hot or black; the assumptions of hotness and blackness are just two stereotypical facts about coffee that we readily take for granted. Lenat and Guha (1990) describe these assumed facts as residing in the *white space* of a text, in the body of common-sense assumptions that are rarely articulated as explicit statements. These culturally-shared common-sense beliefs cannot be harvested directly from a single web resource or document set, but must be gleaned indirectly, from telling phrases that are scattered across the many texts of the web.

Veale and Hao (2007) argue that the most pivotal

reference points of this world-view can be detected in common-place similes like “as lazy as a dog”, “as fat as a hippo” or “as chaste as a nun”. To the extent that this world-view is ingrained in and influenced by how we speak, it can differ from culture to culture and language to language. In English texts, for example, the concept Tortoise is stereotypically associated with the properties *slowness*, *patience* and *wrinkled*, but in Chinese texts, we find that the same animal is a model of *slowness*, *ugliness*, and *nutritional value*. Likewise, because Chinese “wine” has a high alcohol content, the dimension of Strength is much more salient to a Chinese speaker than an English speaker, as reflected in how the word 酒 is used in statements such as 像酒一样浓重, which means “as strong as wine”, or literally, “as wine equally strong”.

In this paper, we compare the same web-based approach to acquiring stereotypical concept descriptions from text using two very different languages, English and Chinese, to determine the extent to which the same cross-cultural knowledge is unearthed for each. In other words, we treat the web as a large parallel corpus (e.g., see Resnick and Smith, 2003), though not of parallel documents in different languages, but of corresponding translation-equivalent phrases. By seeking translation equivalence between different pieces of textually-derived knowledge, this paper addresses the following questions: if a particular syntagmatic pattern is useful for mining knowledge in English, can its translated form be equally useful for Chinese? To what extent does the knowledge acquired using different source languages overlap, and to what extent is this knowledge language- (and culture-) specific? Given that the syntagmatic patterns used in each language are not wholly unambiguous or immune to noise, to what extent should finding the same beliefs expressed in two different languages increase our confidence in the acquired knowledge? Finally, what representational synergies arise from finding these same facts expressed in two different languages?

Given these goals, the rest of the paper assumes the following structure: in section 2, we summarize related work on syntagmatic approaches to knowledge-acquisition; in section 3, we describe our multilingual efforts in English and Chinese to acquire stereotypical or generic-level facts

from the web, by using corresponding translations of the commonplace stereotype-establishing pattern “as ADJ as a NOUN”; and in section 4, we describe how these English and Chinese data-sets can be unified using the bilingual ontology HowNet (Dong and Dong, 2006). This mapping allows us to determine the meaning overlap in both data sets, the amount of noise in each data set, and the degree to which this noise is reduced when parallel translations can be identified. In section 5 we demonstrate the overall usefulness of stereotype-based knowledge-representation by replicating the clustering experiments of Almuhareb and Poesio (2004, 2005) and showing that stereotype-based representations are both compact and predictive of ontological classification. We conclude the paper with some final remarks in section 6.

## 2 Related Work

Text-based approaches to knowledge acquisition range from the ambitiously comprehensive, in which an entire text or resource is fully parsed and analyzed in depth, to the surgically precise, in which highly-specific text patterns are used to eke out correspondingly specific relationships from a large corpus. Endeavors such as that of Harabagiu *et al.* (1999), in which each of the textual glosses in WordNet (Fellbaum, 1998) is linguistically analyzed to yield a sense-tagged logical form, is an example of the former approach. In contrast, foundational efforts such as that of Hearst (1992) typify the latter surgical approach, in which one fishes in a large text for word sequences that strongly suggest a particular semantic relationship, such as hypernymy or, in the case of Charniak and Berland (1999), the part-whole relation. Such efforts offer high precision but low recall, and extract just a tiny (but very useful) subset of the semantic content of a text. The Know-ItAll system of Etzioni *et al.* (2004) employs the same generic patterns as Hearst (e.g., “NPs such as  $NP_1$ ,  $NP_2$ , ...”), and more besides, to extract a whole range of facts that can be exploited for web-based question-answering. Cimiano and Wenderoth (2007) also use a range of Hearst-like patterns to find text sequences in web-text that are indicative of the lexico-semantic properties of words; in particular, these authors use phrases like “to \* a new



NOUN” and “the purpose of NOUN is to \*” to identify the agentive and telic roles of given nouns, thereby fleshing out the noun’s qualia structure as posited by Pustejovsky’s (1990) theory of the generative lexicon.

The basic Hearst approach has even proven useful for identifying the meta-properties of concepts in a formal ontology. Völker *et al.* (2005) show that patterns like “is no longer a|an NOUN” can identify, with reasonable accuracy, those concepts in an ontology that are not rigid, which is to say, concepts like Teacher and Student whose instances may at any point stop being instances of these concepts. Almuhareb and Poesio (2005) use patterns like “a|an|the \* C is|was” and “the \* of the C is|was” to find the actual properties of concepts as they are used in web texts; the former pattern is used to identify value features like *hot*, *red*, *large*, etc., while the latter is used to identify the attribute features that correspond to these values, such as temperature, color and size. Almuhareb and Poesio go on to demonstrate that the values and attributes that are found for word-concepts on the web yield a sufficiently rich representation for these word-concepts to be automatically clustered into a form resembling that assigned by WordNet (see Fellbaum, 1998). Veale and Hao (2007) show that the pattern “as ADJ as a|an NOUN” can also be used to identify the value feature associated with a given concept, and argue that because this pattern corresponds to that of the simile frame in English, the adjectival features that are retrieved are much more likely to be highly salient of the noun-concept (the simile vehicle) that is used. Whereas Almuhareb and Poesio succeed in identifying the range of potential attributes and values that may be possessed by a particular concept, Veale and Hao succeed in identifying the generic properties of a concept as it is conceived in its stereotypical form. As noted by the latter authors, this results in a much smaller yet more diagnostic feature set for each concept. However, because the simile frame is often exploited for ironic purposes in web texts (e.g., “as meaty as a skeleton”), and because irony is so hard to detect, Veale and Hao suggest that the adjective:noun pairings found on the web should be hand-filtered to remove such examples. Given this onerous requirement for hand-filtering, and the unique, culturally-

loaded nature of the noise involved, we use the work of Veale and Hao as the basis for the cross-cultural investigation in this paper.

### 3 Harvesting Knowledge from Similes: English and Chinese

Because similes are containers of culturally-received knowledge, we can reasonably expect the most commonly used similes to vary significantly from language to language, especially when those languages correspond to very different cultures. These similes form part of the linguistic currency of a culture which must be learned by a speaker, and indeed, some remain opaque even to the most educated native speakers. In “A Christmas Carol”, for instance, Dickens (1943/1984) questions the meaning of “as dead as a doornail”, and notes: “I might have been inclined, myself, to regard a coffin-nail as the deadest piece of ironmongery in the trade. But the wisdom of our ancestors is in the simile”.

Notwithstanding the opacity of some instances of the simile form, similes are very revealing about the concepts one most encounters in everyday language. In section 5 we demonstrate that concept descriptions which are harvested from similes are both extremely compact and highly predictive of ontological structure. For now, we turn to the process by which similes can be harvested from the text of the web. In section 3.1 we summarize the efforts of Veale and Hao, whose database of English similes drives part of our current investigation. In section 3.2 we describe how a comparable database of Chinese similes can be harvested from the web.

#### 3.1 Harvesting English Similes

Veale and Hao (2007) use the Google API in conjunction with Princeton WordNet (Fellbaum, 1998) as the basis of their harvesting system. They first extracted a list of antonymous adjectives, such as “hot” or “cold”, from WordNet, the intuition being that explicit similes will tend to exploit properties that occupy an exemplary point on a scale. For every adjective ADJ on this list, they then sent the query “as ADJ as \*” to Google and scanned the first 200 snippets returned for different noun values for the wildcard \*. The complete set of nouns extracted in this way was then used to drive a sec-

ond harvesting phase, in which the query “*as \* as a NOUN*” was used to collect similes that employ different adjectives or which lie beyond the 200-snippet horizon of the original search. Based on this wide-ranging series of core samples (of 200 hits each) from across the web, Veale and Hao report that both phases together yielded 74,704 simile instances (of 42,618 unique types, or unique adjective:noun pairings), relating 3769 different adjectives to 9286 different nouns. As often noted by other authors, such as Völker *et al.* (2005), a pattern-oriented approach to knowledge mining is prone to noise, not least because the patterns used are rarely leak-free (inasmuch as they admit word sequences that do not exhibit the desired relationship), and because these patterns look at small text sequences in isolation from their narrative contexts. Veale and Hao (2007) report that when the above 42,618 simile types are hand-annotated by a native speaker, only 12,259 were judged as non-ironic and meaningful in a null context. In other words, just 29% of the retrieved pairings conform to what one would consider a well-formed and reusable simile that conveys some generic aspect of cultural knowledge. Of those deemed invalid, 2798 unique pairings were tagged as ironic, insofar as they stated precisely the opposite of what is stereotypically believed to be true.

### 3.2 Harvesting Chinese Similes

To harvest a comparable body of Chinese similes from the web, we also use the Google API, in conjunction with both WordNet and HowNet (Dong and Dong, 2006). HowNet is a bilingual lexical ontology that associates English and Chinese word labels with an underlying set of approximately 100,000 lexical concepts. While each lexical concept is defined using a unique numeric identifier, almost all of HowNet’s concepts can be uniquely identified by a pairing of English and Chinese labels. For instance, the word “王八” can mean both Tortoise and Cuckold in Chinese, but the combined label `tortoise|王八` uniquely picks out the first sense while `cuckold|王八` uniquely picks out the second. Though Chinese has a large number of figurative expressions, the yoking of English to Chinese labels still serves to identify the correct sense in almost every case. For instance, “绿帽子” is another word for Cuckold in Chinese, but it can also translate as “green

hat” and “green scarf”. Nonetheless, `green_hat|绿帽子` uniquely identifies the literal sense of “绿帽子” (a green covering) while `green_scarf|绿帽子` and `cuckold|绿帽子` both identify the same human sense, the former being a distinctly culture-specific metaphor for cuckolded males (in English, a dispossessed lover “wears the cuckold’s horns”; in Chinese, one apparently “wears a green scarf”).

We employ the same two-phase design as Veale and Hao: an initial set of Chinese adjectives are extracted from HowNet, with the stipulation that their English translations (as given by HowNet) are also categorized as adjectives in WordNet. We then use the Chinese equivalent of the English simile frame “*像\* 一样ADJ*” (literally, “*as-NOUN-equally-ADJ*”) to retrieve a set of noun values that stereotypically embody these adjectival features. Again, a set of 200 snippets is analyzed for each query, and only those values of the Google *\** wildcard that HowNet categorizes as nouns are accepted. In a second phase, these nouns are used to create new queries of the form “*像Noun一样\**” and the resulting Google snippets are now scanned for adjectival values of *\**.

In all, 25,585 unique Chinese similes (i.e., pairings of an adjective to a noun) are harvested, linking 3080 different Chinese adjectives to 4162 Chinese nouns. When hand-annotated by a native Chinese speaker, the Chinese simile frame reveals itself to be considerably less leaky than the corresponding English frame. Over 58% of these pairings (14,867) are tagged as well-formed and meaningful similes that convey some stereotypical element of world knowledge. The Chinese pattern “*像\*一样\**” is thus almost twice as reliable as the English “*as \* as a \**” pattern. In addition, Chinese speakers exploit the simile frame much less frequently for ironic purposes, since just 185 of the retrieved similes (or 0.7%) are tagged as ironic, compared with ten times as many (or 7%) retrieved English similes. In the next section we consider the extent to which these English and Chinese similes convey the same information.

## 4 Tagging and Mapping of Similes

In each case, the harvesting processes for English and for Chinese allow us to acquire stereotypi-

cal associations between words, not word senses. Nonetheless, the frequent use of synonymous terms introduces a substantial degree of redundancy in these associations, and this redundancy can be used to perform sense discrimination. In the case of English similes, Veale and Hao (2007) describe how two English similes “as A as  $N_1$ ” and “as A as  $N_2$ ” will be mutually disambiguating if  $N_1$  and  $N_2$  are synonyms in WordNet, or if some sense of  $N_1$  is a hypernym or hyponym of some sense of  $N_2$  in WordNet. This heuristic allows Veale and Hao to automatically sense-tag 85%, or 10,378, of the unique similes that are annotated as valid. We apply a similar intuition to the disambiguation of Chinese similes: though HowNet does not support the notion of a synset, different word-senses that have the same meaning will be associated with the same logical definition. Thus, the Chinese word “著名” can translate as “celebrated”, “famous”, “well-known” and “reputable”, but all four of these possible senses, given by celebrated|著名, famous|著名, well-known|著名 and reputable|著名, are associated with the same logical form in HowNet, which defines them as a specialization of ReputationValue|名声值. This allows us to safely identify “著名” with this logical form. Overall, 69% of Chinese similes can have both their adjective and noun assigned to specific HowNet meanings in this way.

#### 4.1 Translation Equivalence Among Similes

Since HowNet represents an integration of English and Chinese lexicons, it can easily be used to connect the English and Chinese data-sets. For while the words used in any given simile are likely to be ambiguous (in the case of one-character Chinese words, highly so), it would seem unlikely that an incorrect translation of a web simile would also be found on the web. This is an intuition that we can now use the annotated data-sets to evaluate.

For every English simile of the form  $\langle A_e \text{ as } N_e \rangle$ , we use HowNet to generate a range of possible Chinese variations  $\langle A_{c0} \text{ as } N_{c0} \rangle$ ,  $\langle A_{c1} \text{ as } N_{c0} \rangle$ ,  $\langle A_{c0} \text{ as } N_{c1} \rangle$ ,  $\langle A_{c1} \text{ as } N_{c1} \rangle$ , ... by using the HowNet lexical entries  $A_e|A_{c0}$ ,  $A_e|A_{c1}$ , ...,  $N_e|N_{c0}$ ,  $N_e|N_{c1}$ , ... as a translation bridge. If the variation  $\langle A_{ci} \text{ as } N_{cj} \rangle$  is found in the Chinese data-set, then translation equivalence is assumed between  $\langle A_e \text{ as}$

<i>Language</i>	<i>Precision</i>	<i>Recall</i>	<i>F1</i>
<i>English</i>	0.76	0.25	0.38
<i>Chinese</i>	0.82	0.27	0.41

Table 1: Automatic filtering of similes using Translation Equivalence.

$N_e \rangle$  and  $\langle A_{ci} \text{ as } N_{cj} \rangle$ ; furthermore,  $A_e|A_{ci}$  is assumed to be the HowNet sense of the adjectives  $A_e$  and  $A_{ci}$  while  $N_{cj}$  is assumed to be the HowNet sense of the nouns  $N_e$  and  $N_{cj}$ . Sense-tagging is thus a useful side-effect of simile-mapping with a bilingual lexicon.

We attempt to find Chinese translation equivalences for all 42,618 of the English adjective:noun pairings harvested by Veale and Hao; this includes both the 12,259 pairings that were hand-annotated as valid stereotypical facts, and the remaining 30,359 that were dismissed as noisy or ironic. Using HowNet, we can establish equivalences from 4177 English similes to 4867 Chinese similes. In those mapped, we find 3194 English similes and 4019 Chinese similes that were hand-annotated as valid by their respective native-speaker judges. In other words, translation equivalence can be used to separate well-formed stereotypical beliefs from ill-formed or ironic beliefs with approximately 80% precision. The precise situation is summarized in Table 1.

As noted in section 3, just 29% of raw English similes and 58% of raw Chinese similes that are harvested from web-text are judged as valid stereotypical statements by a native-speaking judge. For the task of filtering irony and noise from raw data sets, translation equivalence thus offers good precision but poor recall, since most English similes appear not to have a corresponding Chinese variant on the web. Nonetheless, this heuristic allows us to reliably identify a sizeable body of cross-cultural stereotypes that hold in both languages.

##### 4.1.1 Error Analysis

Noisy propositions may add little but empty content to a representation, but ironic propositions will actively undermine a representation from within, leading to inferences that are not just unlikely, but patently false (as is generally the intention of irony). Since Veale and Hao (2007) annotate their data-

set for irony, this allows us to measure the number of egregious mistakes made when using translation equivalence as a simile filter. Overall, we see that 1% of Chinese similes that are accepted via translation equivalence are ironic, accounting for 9% of all errors made when filtering Chinese similes. Likewise, 1% of the English similes that are accepted are ironic, accounting for 5% of all errors made when filtering English similes.

## 4.2 Representational Synergies

By mapping WordNet-tagged English similes onto HowNet-tagged Chinese similes, we effectively obtain two representational viewpoints onto the same shared data set. For instance, though HowNet has a much shallower hierarchical organization than WordNet, it compensates by encapsulating the meaning of different word senses using simple logical formulae of semantic primitives, or sememes, that are derived from the meaning of common Chinese characters. WordNet and HowNet thus offer two complementary levels or granularities of generalization that can be exploited as the context demands.

### 4.2.1 Adjective Organization

Unlike WordNet, HowNet organizes its adjectival senses hierarchically, allowing one to obtain a weaker form of a given description by climbing the hierarchy, or to obtain a stronger form by descending the hierarchy from a particular sense. Thus, one can go up from kaleidoscopic|斑驳陆离 to colored|彩, or down from colored|彩 to any of motley|斑驳, dappled|斑驳, prismatic|斑驳陆离 and even gorgeous|斑斓. Once stereotypical descriptions have been sense-tagged relative to HowNet, they can easily be further enhanced or bleached to suit the context of their use. For example, by allowing a Chinese adjective to denote any of the senses above it or below in the HowNet hierarchy, we can extend the mapping of English to Chinese similes so as to achieve an improved recall of .36 (though we note that this technique reduces the precision of the translation-equivalence heuristic to .75).

As demonstrated by Almuhareb and Poesio (2004), the best conceptual descriptions combine adjectival values with the attributes that they fill.

Because adjectival senses hook into HowNet's upper ontology via a series of abstract taxonyms like TasteValue|美丑值, ReputationValue|名声值 and AmountValue|多少值, a taxonym of the form AttributeValue can be identified for every adjective sense in HowNet. For example, the English adjective "beautiful" can denote either beautiful|美, organized by HowNet under BeautyValue|美丑值, or beautiful|婉, organized by HowNet under gracious|雅 which in turn is organized under GraceValue|典雅值. The adjective "beautiful" can therefore specify either the Grace or Beauty attributes of a concept. Once similes have been sense-tagged, we can build up a picture of most salient attributes of our stereotypical concepts. For instance, "peacock" similes yield the following attributes via HowNet: *Beauty, Appearance, Color, Pride, Behavior, Resplendence, Bearing and Grace*; likewise "demon" similes yield the following: *Morality, Behavior, Temperament, Ability and Competence*.

### 4.2.2 Orthographic Form

The Chinese data-set lacks counterparts to many similes that one would not think of as culturally-determined, such "as red as a ruby", "as cruel as a tyrant" and "as smelly as a skunk". One significant reason for this kind of omission is not cultural difference, but obviousness: many Chinese words are multi-character gestalts of different ideas (see Packard, 2000), so that these ideas form an explicit part of the orthography of a lexical concept. For instance, using HowNet, we can see that skunk|臭鼬 is actually a gestalt of the concepts smelly|臭 and weasel|鼬, so the simile "as smelly as a skunk" is already somewhat redundant in Chinese (somewhat akin to the English similes "as hot as a hotdog" or "as hard as a hardhat").

Such decomposition can allow us to find those English similes that are already orthographically explicit in Chinese word-forms. We simply look for pairs of HowNet senses of the form Noun|XYZ and Adj|X, where X and XYZ are Chinese words and the simile "as Adj as a/an Noun" is found in the English simile set. When we do so, we find that 648 English similes, from "as meaty as a steak" to "as resonant as a cello", are already fossilized in the orthographic realization of the corresponding Chinese concepts. When fossilized similes are uncovered in this way,

the recall of translation equivalence as a noise filter rises to .29, while its precision rises to .84 (see Table 1)

## 5 Empirical Evaluation: Simile-derived Representations

Stereotypes persist in language and culture because they are, more often than not, cognitively useful: by emphasizing the most salient aspects of a concept, a stereotype acts as a dense conceptual description that is easily communicated, widely shared, and which supports rapid inference. To demonstrate the usefulness of stereotype-based concept descriptions, we replicate here the clustering experiments of Almuhareb and Poesio (2004, 2005), who in turn demonstrated that conceptual features that are mined from specific textual patterns can be used to construct WordNet-like ontological structures. These authors used different text patterns for mining feature values (like *hot*) and attributes (like *temperature*), and their experiments evaluated the relative effectiveness of each as a means of ontological clustering. Since our focus in this paper is on the harvesting of feature values, we replicate here only their experiments with values.

Almuhareb and Poesio (2004) used as their experimental basis a sampling of 214 English nouns from 13 of WordNet’s upper-level semantic categories, and proceeded to harvest adjectival features for these noun-concepts from the web using the textual pattern “[a | an | the] \* C [is | was]”. This pattern yielded a combined total of 51,045 value features for these 214 nouns, such as *hot*, *black*, etc., which were then used as the basis of a clustering algorithm in an attempt to reconstruct the WordNet classifications for all 214 nouns. Clustering was performed by the CLUTO-2.1 package (Karypis, 2003), which partitioned the 214 nouns in 13 categories on the basis of their 51,045 web-derived features. Comparing these clusters with the original WordNet-based groupings, Almuhareb and Poesio report a clustering accuracy of 71.96%. In a second, larger experiment, Almuhareb and Poesio (2005) sampled 402 nouns from 21 different semantic classes in WordNet, and harvested 94,989 feature values from the web using the same textual pattern. They then applied the repeated bisections clustering algorithm to

<i>Approach</i>	<i>accuracy</i>	<i>features</i>
<i>Almuhareb + Poesio</i>	71.96%	51,045
<i>Simile-derived stereotypes</i>	70.2%	2,209

Table 2: Results for experiment 1 (214 nouns, 13 WN categories).

<i>Approach</i>	<i>Cluster purity</i>	<i>Cluster entropy</i>	<i>features</i>
<i>Almu. + Poesio</i> (no filtering)	56.7%	38.4%	94,989
<i>Almu. + Poesio</i> (with filtering)	62.7%	33.8%	51345
<i>Simile-derived stereotypes</i> (no filtering)	64.3%	33%	5,547

Table 3: Results for experiment 2 (402 nouns, 21 WN categories).

this larger data set, and report an initial cluster purity measure of 56.7%. Suspecting that a noisy feature set had contributed to the apparent drop in performance, these authors then proceed to apply a variety of noise filters to reduce the set of feature values to 51,345, which in turn leads to an improved cluster purity measure of 62.7%.

We replicated both of Almuhareb and Poesio’s experiments on the same experimental data-sets (of 214 and 402 nouns respectively), using instead the English simile pattern “as \* as a NOUN” to harvest features for these nouns from the web. Note that in keeping with the original experiments, no hand-tagging or filtering of these features is performed, so that every raw match with the simile pattern is used. Overall, we harvest just 2209 feature values for the 214 nouns of experiment 1, and 5547 features for the 402 nouns of experiment 2. A comparison of both sets of results for experiment 1 is shown in Table 2, while a comparison based on experiment 2 is shown in Table 3.

While Almuhareb and Poesio achieve marginally higher clustering on the 214 nouns of experiment 1, they do so by using over 20 times as many features.

In experiment 2, we see a similar ratio of feature quantities before filtering; after some initial filtering, Almuhareb and Poesio reduce their feature set to just under 10 times the size of the simile-derived feature set.

These experiments demonstrate two key points about stereotype-based representations. First, the feature representations do not need to be hand-filtered and noise-free to be effective; we see from the above results that the raw values extracted from the simile pattern prove slightly more effective than filtered feature sets used by Almuhareb and Poesio. Secondly, and perhaps more importantly, stereotype-based representations prove themselves a much more compact means (by factor of 10 to 20 times) of achieving the same clustering goals.

## 6 Conclusions

Knowledge-acquisition from texts can be a process fraught with complexity: such texts - especially web-based texts - are frequently under-determined and vague; highly ambiguous, both lexically and structurally; and dense with figures of speech, hyperbolae and irony. None of the syntagmatic frames surveyed in section 2, from the “NP such as  $NP_1$ ,  $NP_2$  ...” pattern of Hearst (1992) and Etzioni *et al.* (2004) to the “no longer NOUN” pattern of Völker *et al.* (2005), are leak-free and immune to noise. Cimiano and Wenderoth (2007) mitigate this problem somewhat by performing part-of-speech analysis on all extracted text sequences, but the problem remains: the surgical, pattern-based approach offers an efficient and targeted means of knowledge-acquisition from corpora because it largely ignores the context in which these patterns occur; yet one requires this context to determine if a given text sequence really is a good exemplar of the semantic relationship that is sought.

In this paper we have described how stereotypical associations between adjectival properties and noun concepts can be mined from similes in web text. When harvested in both English and Chinese, these associations exhibit two kinds of redundancy that can mitigate the problem of noise. The first kind, *within-language* redundancy, allows us to perform sense-tagging of the adjectives and nouns that are used in similes, by exploiting the

fact that the same stereotypical association can occur in a variety of synonymous forms. By recognizing synonymy between the elements of different similes, we can thus identify the underlying senses (or WordNet synsets) in these similes. The second kind, *between-language* redundancy, exploits the fact that the same associations can occur in different languages, allowing us to exploit translation-equivalence to pin these associations to particular lexical concepts in a multilingual lexical ontology like HowNet. While between-language redundancy is a limited phenomenon, with just 26% of Veale and Hao’s annotated English similes having Chinese translations on the web, this phenomenon does allow us to identify a significant core of shared stereotypical knowledge across these two very different languages.

Overall, our analysis suggests that a comparable number of well-formed Chinese and English similes can be mined from the web (our exploration finds approx. 12,000 unique examples of each). This demonstrates that harvesting stereotypical knowledge from similes is a workable strategy in both languages. Moreover, Chinese simile usage is characterized by two interesting facts that are of some practical import: the simile frame “像NOUN 一样ADJ” is a good deal less leaky and prone to noise than the equivalent English frame, “as ADJ as a NOUN”; and Chinese speakers appear less willing to subvert the stereotypical norms of similes for ironic purposes. Further research is needed to determine whether these observations generalize to other knowledge-mining patterns.

## References

- A. Almuhareb and M. Poesio. 2004. *Attribute-Based and Value-Based Clustering: An Evaluation*. In proceedings of EMNLP 2004, pp 158–165. Barcelona, Spain.
- A. Almuhareb and M. Poesio. 2005. *Concept Learning and Categorization from the Web*. In proceedings of CogSci 2005, the 27th Annual Conference of the Cognitive Science Society. New Jersey: Lawrence Erlbaum.
- C. Dickens. 1843/1981. *A Christmas Carol*. Puffin Books, Middlesex, UK.
- C. Fellbaum. 1998. *WordNet, an electronic lexical database*. MIT Press.
- E. Charniak and M. Berland. 1999. *Finding parts in*

- very large corpora*. In proceedings of the 37th Annual Meeting of the ACL, pp 57-64.
- F. Keller, M. Lapata, and O. Ourioupina. 2002. *Using the web to overcome data sparseness*. In proceedings of EMNLP-02, pp 230-237.
- F. Keller, M. Lapata, and O. Ourioupina. 1990. *Building large knowledge-based systems: representation and inference in the Cyc project*. Addison-Wesley.
- G. Karypis. 2003. *CLUTO: A clustering toolkit*. University of Minnesota.
- J. L. Packard. 2000. *The Morphology of Chinese: A Linguistic and Cognitive Approach*. Cambridge University Press, UK.
- J. Pustejovsky. 1991. *The generative lexicon*. Computational Linguistics 17(4), pp 209-441.
- J. Völker, D. Vrandečić and Y. Sure. 2005. *Automatic Evaluation of Ontologies (AEON)*. In Y. Gil, E. Motta, V. R. Benjamins, M. A. Musen, Proceedings of the 4th International Semantic Web Conference (ISWC2005), volume 3729 of LNCS, pp. 716-731. Springer Verlag Berlin-Heidelberg.
- M. Hearst. 1992. *Automatic acquisition of hyponyms from large text corpora*. In proceedings of the 14th international conference on Computational Linguistics, pp 539-545.
- O. Etzioni, S. Kok, S. Soderland, M. Cafarella, A-M. Popescu, D. Weld, D. Downey, T. Shaked and A. Yates. 2004. *Web-scale information extraction in KnowItAll (preliminary results)*. In proceedings of the 13th WWW Conference, pp 100-109.
- P. Cimiano and J. Wenderoth. 2007. *Automatic Acquisition of Ranked Qualia Structures from the Web*. In proceedings of the 45th Annual Meeting of the ACL, pp 888-895.
- P. Resnik and N. A. Smith. 2003. *The Web as a parallel corpus*. Computational Linguistics, 29(3), pp 349-380.
- S. Harabagiu, G. Miller and D. Moldovan. 1999. *WordNet2 - a morphologically and semantically enhanced resource*. In proceedings of SIGLEX-99, pp 1-8, University of Maryland.
- T. Veale and Y. Hao. 2007. *Making Lexical Ontologies Functional and Context-Sensitive*. In proceedings of the 45th Annual Meeting of the ACL, pp 57-64.
- Z. Dong and Q. Dong. 2006. *HowNet and the Computation of Meaning*. World Scientific: Singapore.

# Semi-supervised Convex Training for Dependency Parsing

**Qin Iris Wang**

Department of Computing Science  
University of Alberta  
Edmonton, AB, Canada, T6G 2E8  
wqin@cs.ualberta.ca

**Dale Schuurmans**

Department of Computing Science  
University of Alberta  
Edmonton, AB, Canada, T6G 2E8  
dale@cs.ualberta.ca

**Dekang Lin**

Google Inc.  
1600 Amphitheatre Parkway  
Mountain View, CA, USA, 94043  
lindek@google.com

## Abstract

We present a novel semi-supervised training algorithm for learning dependency parsers. By combining a supervised large margin loss with an unsupervised least squares loss, a discriminative, convex, semi-supervised learning algorithm can be obtained that is applicable to large-scale problems. To demonstrate the benefits of this approach, we apply the technique to learning dependency parsers from combined labeled and unlabeled corpora. Using a stochastic gradient descent algorithm, a parsing model can be efficiently learned from semi-supervised data that significantly outperforms corresponding supervised methods.

## 1 Introduction

Supervised learning algorithms still represent the state of the art approach for inferring dependency parsers from data (McDonald et al., 2005a; McDonald and Pereira, 2006; Wang et al., 2007). However, a key drawback of supervised training algorithms is their dependence on labeled data, which is usually very difficult to obtain. Perceiving the limitation of supervised learning—in particular, the

heavy dependence on annotated corpora—many researchers have investigated *semi-supervised* learning techniques that can take both labeled and unlabeled training data as input. Following the common theme of “more data is better data” we also use both a limited labeled corpora and a plentiful unlabeled data resource. Our goal is to obtain better performance than a purely supervised approach without unreasonable computational effort. Unfortunately, although significant recent progress has been made in the area of semi-supervised learning, the performance of semi-supervised learning algorithms still fall far short of expectations, particularly in challenging real-world tasks such as natural language parsing or machine translation.

A large number of distinct approaches to semi-supervised training algorithms have been investigated in the literature (Bennett and Demiriz, 1998; Zhu et al., 2003; Altun et al., 2005; Mann and McCallum, 2007). Among the most prominent approaches are self-training, generative models, semi-supervised support vector machines (S3VM), graph-based algorithms and multi-view algorithms (Zhu, 2005).

Self-training is a commonly used technique for semi-supervised learning that has been ap-



plied to several natural language processing tasks (Yarowsky, 1995; Charniak, 1997; Steedman et al., 2003). The basic idea is to bootstrap a supervised learning algorithm by alternating between inferring the missing label information and retraining. Recently, McClosky et al. (2006a) successfully applied self-training to parsing by exploiting available unlabeled data, and obtained remarkable results when the same technique was applied to parser adaptation (McClosky et al., 2006b). More recently, Haffari and Sarkar (2007) have extended the work of Abney (2004) and given a better mathematical understanding of self-training algorithms. They also show connections between these algorithms and other related machine learning algorithms.

Another approach, generative probabilistic models, are a well-studied framework that can be extremely effective. However, generative models use the EM algorithm for parameter estimation in the presence of missing labels, which is notoriously prone to getting stuck in poor local optima. Moreover, EM optimizes a marginal likelihood score that is not discriminative. Consequently, most previous work that has attempted semi-supervised or unsupervised approaches to parsing have not produced results beyond the state of the art supervised results (Klein and Manning, 2002; Klein and Manning, 2004). Subsequently, alternative estimation strategies for unsupervised learning have been proposed, such as *Contrastive Estimation* (CE) by Smith and Eisner (2005). Contrastive Estimation is a generalization of EM, by defining a notion of learner guidance. It makes use of a set of examples (its *neighborhood*) that are similar in some way to an observed example, requiring the learner to move probability mass to a given example, taking only from the example's neighborhood. Nevertheless, CE still suffers from shortcomings, including local minima.

In recent years, SVMs have demonstrated state of the art results in many supervised learning tasks. As a result, many researchers have put effort on developing algorithms for semi-supervised SVMs (S3VMs) (Bennett and Demiriz, 1998; Altun et al., 2005). However, the standard objective of an S3VM is non-convex on the unlabeled data, thus requiring sophisticated global optimization heuristics to obtain reasonable solutions. A number of researchers have proposed several efficient approx-

imation algorithms for S3VMs (Bennett and Demiriz, 1998; Chapelle and Zien, 2005; Xu and Schuurmans, 2005). For example, Chapelle and Zien (2005) propose an algorithm that smoothes the objective with a Gaussian function, and then performs a gradient descent search in the primal space to achieve a local solution. An alternative approach is proposed by Xu and Schuurmans (2005) who formulate a semi-definite programming (SDP) approach. In particular, they present an algorithm for multi-class unsupervised and semi-supervised SVM learning, which relaxes the original non-convex objective into a close convex approximation, thereby allowing a global solution to be obtained. However, the computational cost of SDP is still quite expensive.

Instead of devising various techniques for coping with non-convex loss functions, we approach the problem from a different perspective. We simply replace the non-convex loss on unlabeled data with an alternative loss that is jointly convex with respect to both the model parameters and (the encoding of) the self-trained prediction targets. More specifically, for the loss on the unlabeled data part, we substitute the original unsupervised structured SVM loss with a least squares loss, but keep constraints on the inferred prediction targets, which avoids trivialization. Although using a least squares loss function for classification appears misguided, there is a precedent for just this approach in the early pattern recognition literature (Duda et al., 2000). This loss function has the advantage that the entire training objective on both the labeled and unlabeled data now becomes convex, since it consists of a convex structured large margin loss on labeled data and a convex least squares loss on unlabeled data. As we will demonstrate below, this approach admits an efficient training procedure that can find a global minimum, and, perhaps surprisingly, can systematically improve the accuracy of supervised training approaches for learning dependency parsers.

Thus, in this paper, we focus on *semi-supervised* language learning, where we can make use of both labeled and unlabeled data. In particular, we investigate a semi-supervised approach for structured large margin training, where the objective is a combination of two convex functions, the structured large margin loss on labeled data and the least squares loss on unlabeled data. We apply the result-



Figure 1: A dependency tree

ing semi-supervised convex objective to dependency parsing, and obtain significant improvement over the corresponding supervised structured SVM. Note that our approach is different from the self-training technique proposed in (McClosky et al., 2006a), although both methods belong to semi-supervised training category.

In the remainder of this paper, we first review the supervised structured large margin training technique. Then we introduce the standard semi-supervised structured large margin objective, which is non-convex and difficult to optimize. Next we present a new semi-supervised training algorithm for structured SVMs which is convex optimization. Finally, we apply this algorithm to dependency parsing and show improved dependency parsing accuracy for both Chinese and English.

## 2 Dependency Parsing Model

Given a sentence  $X = (x_1, \dots, x_n)$  ( $x_i$  denotes each word in the sentence), we are interested in computing a directed dependency tree,  $Y$ , over  $X$ . As shown in Figure 1, in a dependency structure, the basic units of a sentence are the syntactic relationships (aka. head-child or governor-dependent or regent-subordinate relations) between two individual words, where the relationships are expressed by drawing links connecting individual words (Manning and Schütze, 1999). The direction of each link points from a head word to a child word, and each word has one and only one head, except for the head of the sentence. Thus a dependency structure is actually a rooted, directed tree. We assume that a directed dependency tree  $Y$  consists of ordered pairs  $(x_i \rightarrow x_j)$  of words in  $X$  such that each word appears in at least one pair and each word has in-degree at most one. Dependency trees are assumed to be projective here, which means that if there is an arc  $(x_i \rightarrow x_j)$ , then  $x_i$  is an ancestor of all the words

between  $x_i$  and  $x_j$ .<sup>1</sup> Let  $\Phi(X)$  denote the set of all the directed, projective trees that span on  $X$ . The parser’s goal is then to find the most preferred parse; that is, a projective tree,  $Y \in \Phi(X)$ , that obtains the highest “score”. In particular, one would assume that the score of a complete spanning tree  $Y$  for a given sentence, whether probabilistically motivated or not, can be decomposed as a sum of local scores for each link (a word pair) (Eisner, 1996; Eisner and Satta, 1999; McDonald et al., 2005a). Given this assumption, the parsing problem reduces to find

$$\begin{aligned}
 Y^* &= \arg \max_{Y \in \Phi(X)} \text{score}(Y|X) \\
 &= \arg \max_{Y \in \Phi(X)} \sum_{(x_i \rightarrow x_j) \in Y} \text{score}(x_i \rightarrow x_j)
 \end{aligned} \tag{1}$$

where the  $\text{score}(x_i \rightarrow x_j)$  can depend on any measurable property of  $x_i$  and  $x_j$  within the sentence  $X$ . This formulation is sufficiently general to capture most dependency parsing models, including probabilistic dependency models (Eisner, 1996; Wang et al., 2005) as well as non-probabilistic models (McDonald et al., 2005a).

For standard scoring functions, particularly those used in non-generative models, we further assume that the score of each link in (1) can be decomposed into a weighted linear combination of features

$$\text{score}(x_i \rightarrow x_j) = \boldsymbol{\theta} \cdot \mathbf{f}(x_i \rightarrow x_j) \tag{2}$$

where  $\mathbf{f}(x_i \rightarrow x_j)$  is a feature vector for the link  $(x_i \rightarrow x_j)$ , and  $\boldsymbol{\theta}$  are the weight parameters to be estimated during training.

## 3 Supervised Structured Large Margin Training

Supervised structured large margin training approaches have been applied to parsing and produce promising results (Taskar et al., 2004; McDonald et al., 2005a; Wang et al., 2006). In particular, structured large margin training can be expressed as minimizing a regularized loss (Hastie et al., 2004), as shown below:

<sup>1</sup>We assume all the dependency trees are projective in our work (just as some other researchers do), although in the real world, most languages are non-projective.

$$\min_{\boldsymbol{\theta}} \frac{\beta}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \sum_i \max_{L_{i,k}} (\Delta(L_{i,k}, Y_i) - \text{diff}(\boldsymbol{\theta}, Y_i, L_{i,k})) \quad (3)$$

where  $Y_i$  is the target tree for sentence  $X_i$ ;  $L_{i,k}$  ranges over all possible alternative  $k$  trees in  $\Phi(X_i)$ ;  $\text{diff}(\boldsymbol{\theta}, Y_i, L_{i,k}) = \text{score}(\boldsymbol{\theta}, Y_i) - \text{score}(\boldsymbol{\theta}, L_{i,k})$ ;  $\text{score}(\boldsymbol{\theta}, Y_i) = \sum_{(x_m \rightarrow x_n) \in Y_i} \boldsymbol{\theta} \cdot \mathbf{f}(x_m \rightarrow x_n)$ , as shown in Section 2; and  $\Delta(L_{i,k}, Y_i)$  is a measure of distance between the two trees  $L_{i,k}$  and  $Y_i$ . This is an application of the structured large margin training approach first proposed in (Taskar et al., 2003) and (Tsochantaridis et al., 2004).

Using the techniques of Hastie et al. (2004) one can show that minimizing the objective (3) is equivalent to solving the quadratic program

$$\begin{aligned} \min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \quad & \frac{\beta}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \mathbf{e}^\top \boldsymbol{\xi} \quad \text{subject to} \\ & \xi_{i,k} \geq \Delta(L_{i,k}, Y_i) - \text{diff}(\boldsymbol{\theta}, Y_i, L_{i,k}) \\ & \xi_{i,k} \geq 0 \\ & \text{for all } i, L_{i,k} \in \Phi(X_i) \end{aligned} \quad (4)$$

where  $\mathbf{e}$  denotes the vector of all 1's and  $\boldsymbol{\xi}$  represents slack variables. This approach corresponds to the training problem posed in (McDonald et al., 2005a) and has yielded the best published results for English dependency parsing.

To compare with the new semi-supervised approach we will present in Section 5 below, we re-implemented the supervised structured large margin training approach in the experiments in Section 7. More specifically, we solve the following quadratic program, which is based on Equation (3)

$$\min_{\boldsymbol{\theta}} \frac{\alpha}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \sum_i \max_L \sum_{m=1}^k \sum_{n=1}^k \Delta(L_{i,m,n}, Y_{i,m,n}) - \text{diff}(\boldsymbol{\theta}, Y_{i,m,n}, L_{i,m,n}) \quad (5)$$

where  $\text{diff}(\boldsymbol{\theta}, Y_{i,m,n}, L_{i,m,n}) = \text{score}(\boldsymbol{\theta}, Y_{i,m,n}) - \text{score}(\boldsymbol{\theta}, L_{i,m,n})$  and  $k$  is the sentence length. We represent a dependency tree as a  $k \times k$  adjacency matrix. In the adjacency matrix, the value of  $Y_{i,m,n}$  is 1 if the word  $m$  is the head of the word  $n$ , 0 otherwise. Since both the distance function  $\Delta(L_i, Y_i)$  and the score function decompose over links, solving (5) is equivalent to solve the original constrained quadratic program shown in (4).

## 4 Semi-supervised Structured Large Margin Objective

The objective of standard semi-supervised structured SVM is a combination of structured large margin losses on both labeled and unlabeled data. It has the following form:

$$\min_{\boldsymbol{\theta}} \frac{\alpha}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \sum_{i=1}^N \text{structured\_loss}(\boldsymbol{\theta}, X_i, Y_i) + \min_{\mathbf{Y}_j} \sum_{j=1}^U \text{structured\_loss}(\boldsymbol{\theta}, X_j, Y_j) \quad (6)$$

where

$$\begin{aligned} \text{structured\_loss}(\boldsymbol{\theta}, X_i, Y_i) \\ = \max_L \sum_{m=1}^k \sum_{n=1}^k \Delta(L_{i,m,n}, Y_{i,m,n}) - \text{diff}(\boldsymbol{\theta}, Y_{i,m,n}, L_{i,m,n}) \end{aligned} \quad (7)$$

$N$  and  $U$  are the number of labeled and unlabeled training sentences respectively, and  $\mathbf{Y}_j$  ranges over guessed targets on the unsupervised data.

In the second term of the above objective shown in (6), both  $\boldsymbol{\theta}$  and  $\mathbf{Y}_j$  are variables. The resulting loss function has a hat shape (usually called hat-loss), which is non-convex. Therefore the objective as a whole is non-convex, making the search for global optimal difficult. Note that the root of the optimization difficulty for S3VMs is the non-convex property of the second term in the objective function. We will propose a novel approach which can deal with this problem. We introduce an efficient approximation—least squares loss—for the structured large margin loss on unlabeled data below.

## 5 Semi-supervised Convex Training for Structured SVM

Although semi-supervised structured SVM learning has been an active research area, semi-supervised structured SVMs have not been used in many real applications to date. The main reason is that most available semi-supervised large margin learning approaches are non-convex or computationally expensive (e.g. (Xu and Schuurmans, 2005)). These techniques are difficult to implement and extremely hard to scale up. We present a semi-supervised algorithm

for structured large margin training, whose objective is a combination of two convex terms: the supervised structured large margin loss on labeled data and the cheap least squares loss on unlabeled data. The combined objective is still convex, easy to optimize and much cheaper to implement.

### 5.1 Least Squares Convex Objective

Before we introduce the new algorithm, we first introduce a convex loss which we apply it to unlabeled training data for the semi-supervised structured large margin objective which we will introduce in Section 5.2 below. More specifically, we use a *structured* least squares loss to approximate the structured large margin loss on unlabeled data. The corresponding objective is:

$$\min_{\boldsymbol{\theta}, \mathbf{Y}_j} \frac{\alpha}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \frac{\lambda}{2} \sum_{j=1}^U \sum_{m=1}^k \sum_{n=1}^k \left( \boldsymbol{\theta}^\top \mathbf{f}(X_{j,m} \rightarrow X_{j,n}) - Y_{j,m,n} \right)^2 \quad (8)$$

subject to constraints on  $\mathbf{Y}$  (explained below).

The idea behind this objective is that for each possible link ( $X_{j,m} \rightarrow X_{j,n}$ ), we intend to minimize the difference between the link and the corresponding estimated link based on the learned weight vector. Since this is conducted on unlabeled data, we need to estimate both  $\boldsymbol{\theta}$  and  $\mathbf{Y}_j$  to solve the optimization problem. As mentioned in Section 3, a dependency tree  $\mathbf{Y}_j$  is represented as an adjacency matrix. Thus we need to enforce some constraints in the adjacency matrix to make sure that each  $\mathbf{Y}_j$  satisfies the dependency tree constraints. These constraints are critical because they prevent (8) from having a trivial solution in  $\mathbf{Y}$ . More concretely, suppose we use rows to denote heads and columns to denote children. Then we have the following constraints on the adjacency matrix:

- (1) All entries in  $\mathbf{Y}_j$  are between 0 and 1 (convex relaxation of discrete directed edge indicators);
- (2) The sum over all the entries on each column is equal to one (one-head rule);
- (3) All the entries on the diagonal are zeros (no self-link rule);

- (4)  $Y_{j,m,n} + Y_{j,n,m} \leq 1$  (anti-symmetric rule), which enforces directedness.

One final constraint that is sufficient to ensure that a directed tree is obtained, is connectedness (i.e. acyclicity), which can be enforced with an additional semidefinite constraint. Although convex, this constraint is more expensive to enforce, therefore we drop it in our experiments below. (However, adding the semidefinite connectedness constraint appears to be feasible on a sentence by sentence level.)

Critically, the objective (8) is *jointly* convex in both the weights  $\boldsymbol{\theta}$  and the edge indicator variables  $\mathbf{Y}$ . This means, for example, that there are no local minima in (8)—*any* iterative improvement strategy, if it converges at all, must converge to a global minimum.

### 5.2 Semi-supervised Convex Objective

By combining the convex structured SVM loss on labeled data (shown in Equation (5)) and the convex least squares loss on unlabeled data (shown in Equation (8)), we obtain a semi-supervised structured large margin loss

$$\min_{\boldsymbol{\theta}, \mathbf{Y}_j} \frac{\alpha}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \sum_{i=1}^N \text{structured\_loss}(\boldsymbol{\theta}, X_i, Y_i) + \sum_{j=1}^U \text{least\_squares\_loss}(\boldsymbol{\theta}, X_j, Y_j) \quad (9)$$

subject to constraints on  $\mathbf{Y}$  (explained above).

Since the summation of two convex functions is also convex, so is (9). Replacing the two losses with the terms shown in Equation (5) and Equation (8), we obtain the final convex objective as follows:

$$\min_{\boldsymbol{\theta}, \mathbf{Y}_j} \frac{\alpha}{2N} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \sum_{i=1}^N \max_L \sum_{m=1}^k \sum_{n=1}^k \Delta(L_{i,m,n}, Y_{i,m,n}) - \text{diff}(\boldsymbol{\theta}, Y_{i,m,n}, L_{i,m,n}) + \frac{\alpha}{2U} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \frac{\lambda}{2} \sum_{j=1}^U \sum_{m=1}^k \sum_{n=1}^k \left( \boldsymbol{\theta}^\top \mathbf{f}(X_{j,m} \rightarrow X_{j,n}) - Y_{j,m,n} \right)^2 \quad (10)$$

subject to constraints on  $\mathbf{Y}$  (explained above), where  $\text{diff}(\boldsymbol{\theta}, Y_{i,m,n}, L_{i,m,n}) = \text{score}(\boldsymbol{\theta}, Y_{i,m,n}) -$

$score(\theta, L_{i,m,n})$ ,  $N$  and  $U$  are the number of labeled and unlabeled training sentences respectively, as we mentioned before. Note that in (10) we have split the regularizer into two parts; one for the supervised component of the objective, and the other for the unsupervised component. Thus the semi-supervised convex objective is regularized proportionally to the number of labeled and unlabeled training sentences.

## 6 Efficient Optimization Strategy

To solve the convex optimization problem shown in Equation (10), we used a gradient descent approach which simply uses stochastic gradient steps. The procedure is as follows.

- Step 0, initialize the  $\mathbf{Y}_j$  variables of each unlabeled sentence as a right-branching (left-headed) chain model, i.e. the head of each word is its left neighbor.
- Step 1, pass through all the labeled training sentences one by one. The parameters  $\theta$  are updated based on each labeled sentence.
- Step 2, based on the learned parameter weights from the labeled data, update  $\theta$  and  $\mathbf{Y}_j$  on each unlabeled sentence alternatively:
  - treat  $\mathbf{Y}_j$  as a constant, update  $\theta$  on each unlabeled sentence by taking a local gradient step;
  - treat  $\theta$  as a constant, update  $\mathbf{Y}_j$  by calling the optimization software package CPLEX to solve for an optimal local solution.
- Repeat the procedure of step 1 and step 2 until maximum iteration number has reached.

This procedure works efficiently on the task of training a dependency parser. Although  $\theta$  and  $\mathbf{Y}_j$  are updated locally on each sentence, progress in minimizing the total objective shown in Equation (10) is made in each iteration. In our experiments, the objective usually converges within 30 iterations.

## 7 Experimental Results

Given a convex approach to semi-supervised structured large margin training, and an efficient training

algorithm for achieving a global optimum, we now investigate its effectiveness for dependency parsing. In particular, we investigate the accuracy of the results it produces. We applied the resulting algorithm to learn dependency parsers for both English and Chinese.

### 7.1 Experimental Design

#### Data Sets

Since we use a semi-supervised approach, both labeled and unlabeled training data are needed. For experiment on English, we used the English Penn Treebank (PTB) (Marcus et al., 1993) and the constituency structures were converted to dependency trees using the same rules as (Yamada and Matsumoto, 2003). The standard training set of PTB was split into 2 parts: labeled training data—the first 30k sentences in section 2-21, and unlabeled training data—the remaining sentences in section 2-21. For Chinese, we experimented on the Penn Chinese Treebank 4.0 (CTB4) (Palmer et al., 2004) and we used the rules in (Bikel, 2004) for conversion. We also divided the standard training set into 2 parts: sentences in section 400-931 and sentences in section 1-270 are used as labeled and unlabeled data respectively. For both English and Chinese, we adopted the standard development and test sets throughout the literature.

As listed in Table 1 with greater detail, we experimented with sets of data with different sentence length: PTB-10/CTB4-10, PTB-15/CTB4-15, PTB-20/CTB4-20, CTB4-40 and CTB4, which contain sentences with up to 10, 15, 20, 40 and all words respectively.

#### Features

For simplicity, in current work, we only used two sets of features—word-pair and tag-pair indicator features, which are a subset of features used by other researchers on dependency parsing (McDonald et al., 2005a; Wang et al., 2007). Although our algorithms can take arbitrary features, by only using these simple features, we already obtained very promising results on dependency parsing using both the supervised and semi-supervised approaches. Using the full set of features described in (McDonald et al., 2005a; Wang et al., 2007) and comparing the corresponding dependency parsing

English	PTB-10	Training(l/ul) Dev Test	3026/1016 163 270
	PTB-15	Training Dev Test	7303/2370 421 603
	PTB-20	Training Dev Test	12519/4003 725 1034
Chinese	CTB4-10	Training(l/ul) Dev Test	642/347 61 40
	CTB4-15	Training Dev Test	1262/727 112 83
	CTB4-20	Training Dev Test	2038/1150 163 118
	CTB4-40	Training Dev Test	4400/2452 274 240
	CTB4	Training Dev Test	5314/2977 300 289

Table 1: Size of Experimental Data (# of sentences)

results with previous work remains a direction for future work.

### Dependency Parsing Algorithms

For simplicity of implementation, we use a standard CKY parser in the experiments, although Eisner’s algorithm (Eisner, 1996) and the Spanning Tree algorithm (McDonald et al., 2005b) are also applicable.

### 7.2 Results

We evaluate parsing accuracy by comparing the directed dependency links in the parser output against the directed links in the treebank. The parameters  $\alpha$  and  $\lambda$  which appear in Equation (10) were tuned on the development set. Note that, during training, we only used the raw sentences of the unlabeled data. As shown in Table 2 and Table 3, for each data set, the semi-supervised approach achieves a significant improvement over the supervised one in dependency parsing accuracy on both Chinese and English. These positive results are somewhat surprising since a very simple loss function was used on

Training	Test length	Supervised	Semi-sup
Train-10	$\leq 10$	82.98	<b>84.50</b>
Train-15	$\leq 10$	84.80	<b>86.93</b>
	$\leq 15$	76.96	<b>80.79</b>
Train-20	$\leq 10$	84.50	<b>86.32</b>
	$\leq 15$	78.77	<b>80.57</b>
	$\leq 20$	74.89	<b>77.85</b>
Train-40	$\leq 10$	84.19	<b>85.71</b>
	$\leq 15$	78.03	<b>81.21</b>
	$\leq 20$	76.25	<b>77.79</b>
	$\leq 40$	68.17	<b>70.90</b>
Train-all	$\leq 10$	82.67	<b>84.80</b>
	$\leq 15$	77.92	<b>79.30</b>
	$\leq 20$	77.30	77.24
	$\leq 40$	70.11	<b>71.90</b>
	all	66.30	<b>67.35</b>

Table 2: Supervised and Semi-supervised Dependency Parsing Accuracy on Chinese (%)

Training	Test length	Supervised	Semi-sup
Train-10	$\leq 10$	87.77	<b>89.17</b>
Train-15	$\leq 10$	88.06	<b>89.31</b>
	$\leq 15$	81.10	<b>83.37</b>
Train-20	$\leq 10$	88.78	<b>90.61</b>
	$\leq 15$	83.00	<b>83.87</b>
	$\leq 20$	77.70	<b>79.09</b>

Table 3: Supervised and Semi-supervised Dependency Parsing Accuracy on English (%)

the unlabeled data. A key benefit of the approach is that a straightforward training algorithm can be used to obtain global solutions. Note that the results of our model are not directly comparable with previous parsing results shown in (McClosky et al., 2006a), since the parsing accuracy is measured in terms of dependency relations while their results are  $f$ -score of the bracketings implied in the phrase structure.

## 8 Conclusion and Future Work

In this paper, we have presented a novel algorithm for semi-supervised structured large margin training. Unlike previous proposed approaches, we introduce a convex objective for the semi-supervised learning algorithm by combining a convex structured SVM loss and a convex least square loss. This new semi-supervised algorithm is much more computationally efficient and can easily scale up. We have proved our hypothesis by applying the algorithm to the significant task of dependency parsing. The experimental results show that the proposed semi-supervised large margin training algorithm outperforms the supervised one, without much additional computational cost.

There remain many directions for future work. One obvious direction is to use the whole Penn Treebank as labeled data and use some other unannotated data source as unlabeled data for semi-supervised training. Next, as we mentioned before, a much richer feature set can be used in our model to get better dependency parsing results. Another direction is to apply the semi-supervised algorithm to other natural language problems, such as machine translation, topic segmentation and chunking. In these areas, there are only limited annotated data available. Therefore semi-supervised approaches are necessary to achieve better performance. The proposed semi-supervised convex training approach can be easily applied to these tasks.

## Acknowledgments

We thank the anonymous reviewers for their useful comments. Research is supported by the Alberta Ingenuity Center for Machine Learning, NSERC, MITACS, CFI and the Canada Research Chairs program. The first author was also funded by the Queen Elizabeth II Graduate Scholarship.

## References

- S. Abney. 2004. Understanding the yarowsky algorithm. *Computational Linguistics*, 30(3):365–395.
- Y. Altun, D. McAllester, and M. Belkin. 2005. Maximum margin semi-supervised learning for structured variables. In *Proceedings of Advances in Neural Information Processing Systems 18*.
- K. Bennett and A. Demiriz. 1998. Semi-supervised support vector machines. In *Proceedings of Advances in Neural Information Processing Systems 11*.
- D. Bikel. 2004. Intricacies of Collins’ parsing model. *Computational Linguistics*, 30(4).
- O. Chapelle and A. Zien. 2005. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*.
- E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 598–603.
- R. Duda, P. Hart, and D. Stork. 2000. *Pattern Classification*. Wiley, second edition.
- J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the International Conference on Computational Linguistics*.
- G. Haffari and A. Sarkar. 2007. Analysis of semi-supervised learning with the yarowsky algorithm. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
- T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. 2004. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415.
- D. Klein and C. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- D. Klein and C. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- G. S. Mann and A. McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of International Conference on Machine Learning*.
- C. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

- M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- D. McClosky, E. Charniak, and M. Johnson. 2006a. Effective self-training for parsing. In *Proceedings of the Human Language Technology: the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.
- D. McClosky, E. Charniak, and M. Johnson. 2006b. Reranking and self-training for parser adaptation. In *Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics*.
- R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of European Chapter of the Annual Meeting of the Association for Computational Linguistics*.
- R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technologies and Conference on Empirical Methods in Natural Language Processing*.
- M. Palmer *et al.* 2004. *Chinese Treebank 4.0*. Linguistic Data Consortium.
- N. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of the European Chapter of the Annual Meeting of the Association for Computational Linguistics*, pages 331–338.
- B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Proceedings of Advances in Neural Information Processing Systems 16*.
- B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of International Conference on Machine Learning*.
- Q. Wang, D. Schuurmans, and D. Lin. 2005. Strictly lexical dependency parsing. In *Proceedings of the International Workshop on Parsing Technologies*, pages 152–159.
- Q. Wang, C. Cherry, D. Lizotte, and D. Schuurmans. 2006. Improved large margin dependency parsing via local constraints and Laplacian regularization. In *Proceedings of The Conference on Computational Natural Language Learning*, pages 21–28.
- Q. Wang, D. Lin, and D. Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1756–1762.
- L. Xu and D. Schuurmans. 2005. Unsupervised and semi-supervised multi-class support vector machines. In *Proceedings the Association for the Advancement of Artificial Intelligence*.
- H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the International Workshop on Parsing Technologies*.
- D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts.
- X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of International Conference on Machine Learning*.
- X. Zhu. 2005. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison.



# Chinese-English Backward Transliteration Assisted with Mining Monolingual Web Pages

Fan Yang, Jun Zhao, Bo Zou, Kang Liu, Feifan Liu

National Laboratory of Pattern Recognition  
Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

{fyang, jzhao, bzou, kliu, ffliu}@nlpr.ia.ac.cn

## Abstract

In this paper, we present a novel backward transliteration approach which can further assist the existing statistical model by mining monolingual web resources. Firstly, we employ the syllable-based search to revise the transliteration candidates from the statistical model. By mapping all of them into existing words, we can filter or correct some pseudo candidates and improve the overall recall. Secondly, an AdaBoost model is used to re-rank the revised candidates based on the information extracted from monolingual web pages. To get a better precision during the re-ranking process, a variety of web-based information is exploited to adjust the ranking score, so that some candidates which are less possible to be transliteration names will be assigned with lower ranks. The experimental results show that the proposed framework can significantly outperform the baseline transliteration system in both precision and recall.

## 1 Introduction \*

The task of Name Entity (NE) translation is to translate a name entity from source language to target language, which plays an important role in machine translation and cross-language information retrieval (CLIR). Transliteration is a subtask in NE translation, which translates NEs based on the phonetic similarity. In NE translation, most person names are transliterated, and some parts of location names or organization names also need to be transliterated. Transliteration has two directions: forward transliteration which transforms an original name into target language, and backward transliteration which recovers a name back to its original expression. For instance, the original English per-

son name “Clinton” can be forward transliterated to its Chinese expression “克/ke 林/lin 顿/dun” and the backward transliteration is the inverse processing. In this paper, we focus on backward transliteration from Chinese to English.

Many previous researches have tried to build a transliteration model using statistical approach [Knight and Graehl, 1998; Lin and Chen, 2002; Virga and Khudanpur, 2003; Gao, 2004]. There are two main challenges in statistical backward transliteration: First, statistical transliteration approach selects the most probable translations based on the knowledge learned from the training data. This approach, however, does not work well when there are multiple standards [Gao, 2004]. Second, backward transliteration is more challenging than forward transliteration as it is required to disambiguate the noises introduced in the forward transliteration and estimate the original name as close as possible [Lin and Chen, 2002]. One of the most important causes in introducing noises is that: some silent syllables in original names have been missing when they are transliterated to target language. For example, when “Campbell” is transliterated into “坎/kan 贝/bei 尔/er”, the “p” is missing.

In order to make up the disadvantages of statistical approach, some researchers have been seeking for the assistance of web resource. [Wang et al., 2004; Cheng et al., 2004; Nagata et al., 2001; Zhang et al, 2005] used bilingual web pages to extract translation pairs. Other efforts have been made to combine a statistical transliteration model with web mining [Al-Onaizan and Knight, 2002; Long Jiang et al, 2007]. Most of these methods need bilingual resources. However, those kinds of resources are not readily available in many cases. Moreover, to search for bilingual pages, we have to depend on the performance of search engines. We can't get Chinese-English bilingual pages when the input is a Chinese query. Therefore, the existing

\*Contact: Jun ZHAO, [jzhao@nlpr.ia.ac.cn](mailto:jzhao@nlpr.ia.ac.cn).

assistance approaches using web-mining to assist transliteration are not suitable for Chinese to English backward transliteration.

Thus in this paper, we mainly focus on the following two problems to be solved in transliteration.

**Problem I:** Some silent syllables are missing in English-Chinese forward transliteration. How to recover them effectively and efficiently in backward transliteration is still an open problem.

**Problem II:** Statistical transliteration always chooses the translations based on probabilities. However, in some cases, the correct translation may have lower probability. Therefore, more studies are needed on combination with other techniques as supplements.

Aiming at these two problems, we propose a method which mines monolingual web resources to assist backward transliteration. The main ideas are as follows. We assume that for every Chinese entity name which needs to be backward transliterated to an English original name, the correct transliteration exists somewhere in the web. What we need to do is to find out the answers based on the clues given by statistical transliteration results. Different from the traditional methods which extract transliteration pairs from bilingual pages, we only use monolingual web resources. Our method has two advantages. Firstly, there are much more monolingual web resources available to be used. Secondly, our method can revise the transliteration candidates to the existing words before the subsequent re-ranking process, so that we can better mine the correct transliteration from the Web.

Concretely, there are two phases involved in our approach. In the first phase, we split the result of transliteration into syllables, and then a syllable-based searching processing can be employed to revise the result in a word list generated from web pages, with an expectation of higher recall of transliteration. In the second phase, we use a revised word as a search query to get its contexts and hit information, which are integrated into the AdaBoost classifier to determine whether the word is a transliteration name or not with a confidence score. This phase can readjust the candidate's score to a more reasonable point so that precision of transliteration can be improved. Table 1 illustrates how to transliterate the Chinese name “阿/a加/jia西/xi” back to “Agassi”.

Chinese name	Transliteration results	Revised Candidate	Re-rank Results
阿加西	aggasi agahi agacy agacie ...	aggasi agahi agacy agacie ...	aggasi agahi agacy agacie ...

阿加西	aggasi	agasi	<b>agassi</b>
a jia xi	agahi	agathi	agasi
Agassi	agacy	agathe	agache
	agacie	<b>agassi</b>	agga
	...	...	...

Table 1. An example of transliteration flow

The experimental results show that our approach improves the recall from 41.73% to 59.28% in open test when returning the top-100 results, and the top-5 precision is improved from 19.69% to 52.19%.

The remainder of the paper is structured as follows. Section 2 presents the framework of our system. We discuss the details of our statistical transliteration model in Section 3. In Section 4, we introduce the approach of revising and re-ranking the results of transliteration. The experiments are reported in Section 5. The last section gives the conclusion and the prediction of future work.

## 2 System Framework

Our system has three main modules.

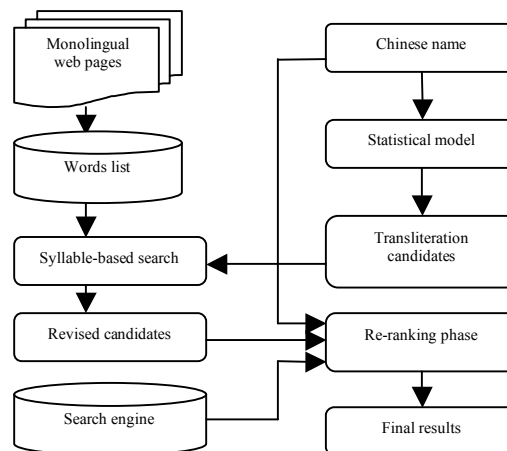


Figure 1. System framework

1) **Statistical transliteration:** This module receives a Chinese Pinyin sequence as its input, and output the  $N$ -best results as the transliteration candidates.

2) **Candidate transliteration revision through syllable-based searching:** In the module, a transliteration candidate is transformed into a syllable query. We use a syllable-based searching strategy to select the revised candidate from a huge word list. Each word in the list is indexed by syllables, and the similarity between the word and the query is calculated. The most similar words are returned as the revision results. This module guar-

antees the transliteration candidates are all existing words.

3) **Revised candidate re-ranking in web pages**: In the module, we search the revised candidates to get their contexts and hit information which we can use to score the probability of being a transliteration name. This phase doesn't generate new candidates, but re-rank the revised candidate set to improve the performance in top-5.

Under this framework, we can solve the two problems of statistical model mentioned above.

(1) The silent syllables will be given lower weights in syllable-based search, so the missing syllables will be recovered through selecting the most similar existing words which can contain some silent syllables.

(2) The query expansion technology can recall more potential transliteration candidates by expanding syllables to their "synonymies". So the mistakes introduced when selecting syllables in statistical transliteration will be corrected through giving suitable weights to synonymies.

Through the revision phase, the results of statistical model which may have illegal spelling will be mapped to its most similar existing words. That can improve the recall. In re-ranking phase, the revised candidate set will be re-ranked to put the right answer on the top using hybrid information got from web resources. So the precision of transliteration will be improved.

### 3 Statistical Transliteration Model

We use syllables as translation units to build a statistical Chinese-English backward transliteration model in our system.

#### 3.1 Traditional Statistical Translation Model

[P. Brown et al., 1993] proposed an IBM source-channel model for statistical machine translation (SMT). When the channel output  $f=f_1, f_2, \dots, f_n$  observed, we use formula (1) to seek for the original sentence  $e=e_1, e_2, \dots, e_n$  with the most likely posteriori.

$$e' = \arg \max_e P(e | f) = \arg \max_e P(f | e)P(e) \quad (1)$$

The translation model  $P(f | e)$  is estimated from a paired corpus of foreign-language sentences and their English translations. The language model  $P(e)$  is trained from English texts.

#### 3.2 Our Transliteration Model

The alignment method is the base of statistical transliteration model. There are mainly two kinds of alignment methods: phoneme-based alignment [Knight and Graehl, 1998; Virga and Khudanpur, 2003] and grapheme-based alignment [Long Jiang, 2007]. In our system, we adopt the syllable-based alignment from Chinese pinyin to English syllables, where the syllabication rules mentioned in [Long Jiang et al., 2007] are used.

For example, Chinese name "希/xi 尔/er 顿/dun" and its backward transliteration "Hilton" can be aligned as follows. "Hilton" is split into syllable sequence as "hi/l/ton", and the alignment pairs are "xi-hi", "er-l", "dun-ton".

Based on the above alignment method, we can get our statistical Chinese-English backward transliteration model as,

$$E = \arg \max_E p(PY | ES)p(ES) \quad (2)$$

Where,  $PY$  is a Chinese Pinyin sequence,  $ES$  is a English syllables sequence,  $p(PY | ES)$  is the probability of translating  $ES$  into  $PY$ ,  $p(ES)$  is the generative probability of a English syllable language model.

#### 3.3 The Difference between Backward Transliteration and Traditional Translation

Chinese-English backward transliteration has some differences from traditional translation.

1) We don't need to adjust the order of syllables when transliteration.

2) The language model in backward transliteration describes the relationship of syllables in words. It can't work as well as the language model describing the word relationship in sentences.

We think that the crucial problem in backward transliteration is selecting the right syllables at every step. It's very hard to obtain the exact answer only based on the statistical transliteration model. We will try to improve the statistical model performance with the assistance of mining web resources.

#### 4 Mining Monolingual Web Pages to Assist Backward Transliteration

In order to get assistance from monolingual Web resource to improve statistical transliteration, our

method contains two main phases: “revision” and “re-ranking”. In the revision phase, transliteration candidates are revised using syllable-based search in the word list, which are generated by collecting the existing words in web pages. Because the process of named entity recognition may lose some NEs, we will reserve all the words in web corpus without any filtering. The revision process can improve the recall through correcting some mistakes in the transliteration results of statistical model.

In the re-ranking phase, we search every revised candidate on English pages, score them according to their contexts and hit information so that the right answer will be given a higher rank.

#### 4.1 Using Syllable-based Retrieval to Revise Transliteration Candidates

In this section, we will propose two methods respectively for the two problems of statistical model mentioned in section 1.

##### 4.1.1 Syllable-based retrieval model

When we search a transliteration candidate  $t_i$  in the word list, we firstly split it into syllables  $\{es_1, es_2, \dots, es_n\}$ . Then this syllable sequence is used as a query for syllable-based searching.

We define some notions here.

- Term set  $T = \{t_1, t_2, \dots, t_k\}$  is an orderly set of all syllables which can be viewed as terms.
- Pinyin set  $P = \{py_1, py_2, \dots, py_k\}$  is an orderly set of all Pinyin.
- An input word can be represented by a vector of syllables  $\{es_1, es_2, \dots, es_n\}$ .

We calculate the similarity between a transliteration result and each word in the list to select the most similar words as the revised candidates. The  $\{es_1, es_2, \dots, es_n\}$  will be transformed into a vector  $V_{query} = \{t_1, t_2, \dots, t_k\}$  where  $t_i$  represents the  $i$ th term in  $T$ . The value of  $t_i$  is equal to 0 if the  $i$ th term doesn't appear in query. In the same way, the word in list can also be transformed into vector representation. So the similarity can be calculated as the inner product between these two vectors.

We don't use *tf* and *idf* conceptions as traditional information retrieval (IR) to calculate the terms' weight. We use the weight of  $t_i$  to express the expectation probability of  $i$ th term having pronunciation. If the term has a lower probability of having pronunciation, its weight is low. So when we searching, the missing silent syllables in the results

of statistical transliteration model can be recovered because such syllables have little impact on similarity measurement. The formula we used is as follows.

$$Sim(query, word) = \frac{V_{query} \times V_{word}}{L_{word} / L_{py}} \quad (3)$$

The numerator is the inner product of two vectors. The denominator is the length of word  $L_{word}$  divided by the length of Chinese pinyin sequence  $L_{py}$ . In this formula, the more syllables in one word, the higher score of inner production it may get, but the word will get a loss for its longer length. The word which has the shortest length and the highest syllable hitting ratio will be the best.

Another difference from traditional IR is how to deal with the order of the words in a query. According to transliteration, the similarity must be calculated under the limitation of keeping order, which can't be satisfied by current methods. We use the algorithm like calculating the edit distance between two words. The syllables are viewed as the units which construct a word. The edit distance calculation finds the best matching with the least operation cost to change one word to another word by using deletion/addition/insertion operations on syllables. But the complexity will be too high to afford if we calculate the edit distance between a query and each word in the list. So, we just calculate the edit distance for the words which get high score without the order limitation. This trade off method can save much time but still keep performance.

##### 4.1.2 Mining the Equivalent through Syllable Expansion

In most collections, the same concept may be referred to using different words. This issue, known as synonymy, has an impact on the recall of most information retrieval systems. In this section, we try to use the expansion technology to solve problem II. There are three kinds of expansions to be explained below.

**Syllable expansion based on phonetic similarity:** The syllables which correspond to the same Chinese pinyin can be viewed as synonymies. For example, the English syllables “din” and “tin” can be aligned to the same Chinese pinyin “ding”.

Given a Chinese pinyin sequence  $\{py_1, py_2, \dots, py_n\}$  as the input of transliteration model, for every  $py_i$ , there are a set of syllables

$\{es_1, es_2, \dots, es_k\}$  which can be selected as its translation. The statistical model will select the most probable one, while others containing the right answer are discarded. To solve this problem, we expand the query to take the synonymies of terms into consideration. We create an expansion set for each Chinese pinyin. A syllable  $es_i$  will be selected into the expansion set of  $py_j$  based on the alignment probability  $P(es_i|py_j)$  which can be extracted from the training corpus. The phonetic similarity expansion is based on the input Chinese Pinyin sequence, so it's same for all candidates.

**Syllable expansion based on syllable similarity:** If two syllables have similar alignment probability with every pinyin, we can view these two syllables as synonymy. Therefore, if a syllable is in the query, its synonymies should be contained too. For example, “fea” and “fe” can replace each other.

To calculate the similarity, we first obtain the alignment probability  $P(py_i|es_k)$  of every syllable. Then the distance between any two syllables will be calculated using formula (4).

$$Sim(es_j, es_k) = \frac{1}{N} \sum_{i=1}^N P(py_i | es_j) P(py_i | es_k) \quad (4)$$

This formula is used to evaluate the similarity of two syllables in alignment. The expansion set of the  $i$ th syllable can be generated by selecting the most similar  $N$  syllables. This kind of expansion is conducted upon the output of statistical transliteration model.

**Syllable expansion based on syllable edit distance:** The disadvantage of last two expansions is that they are entirely dependent on the training set. In other word, if some syllables haven't appeared in the training corpus, they will not be expanded. To solve the problem, we use the method of expansion based on edit distance. We use edit distance to measure the similarity between two syllables, one is in training set and the other is absent. Because the edit distance expansion is not very relevant to pronunciation, we will give this expansion method a low weight in combination. It works when new syllables arise.

**Combine the above three strategies:** We will combine the three kinds of expansion method together. We use the linear interpolation to integrate them. The formulas are follows.

$$S = (1 - \alpha)S_{pre} + \alpha S_{sy} + \beta S_{ed} \quad (5)$$

$$S = (1 - \alpha)S_{pre} + \alpha S_{py} + \beta S_{ed} \quad (6)$$

where  $S_{pre}$  is the score of exact matching,  $S_{sy}$  is the score of expansion based on syllables similarity and  $S_{py}$  based on phonetic similarity. We will adjust these parameters to get the best performance. The experimental results and analysis will be reported in section 5.3.

## 4.2 Re-Ranking the Revised Candidates Set using the Monolingual Web Resource

In the first phase, we have generated the revised candidate set  $\{rc_1, rc_2, \dots, rc_n\}$  from the word list using the transliteration results as clues. The objective is to improve the overall recall. In the second phase, we try to improve the precision, i.e. we wish to re-rank the candidate set so that the correct answer will be put in a higher rank.

[Al-Onaizan et al., 2002] has proposed some methods to re-score the transliteration candidates. The limitation of their approach is that some candidates are propbale not existing words, with which we will not get any information from web. So it can only re-rank the transliteration results to improve the precision of top-5. In our work, we can improve the recall of transliteration through the revising process before re-ranking.

In this section, we employ the AdaBoost framework which integrates several kinds of features to re-rank the revised candidate set. The function of the AdaBoost classifier is to calculate the probability of the candidate being a NE. Then we can re-rank the revised candidate set based on the score. The features used in our system are as follows.

**NE or not:** Using  $rc_i$  as query to search for monolingual English Web Pages, we can get the context set  $\{T_{i1}, T_{i2}, \dots, T_{in}\}$  of  $rc_i$ . Then for every  $T_{ik}$ , we use the named entity recognition (NER) software to determine whether  $rc_i$  is a NE or not. If  $rc_i$  is recognized as a NE in some  $T_{ik}$ ,  $rc_i$  will get a score. If  $rc_i$  can't be recognized as NE in any contexts, it will be pruned.

**The hit of the revised candidate:** We can get the hit information of  $rc_i$  from search engine. It is used to evaluate the importance of  $rc_i$ . Unlike [Al-Onaizan et al., 2002], in which the hit can be used to eliminate the translation results which contain illegal spelling, we just use hit number as a feature.

**The limitation of compound NEs:** When transliterating a compound NE, we always split them into several parts, and then combine their transliteration results together. But in this circumstance,

every part can add a limitation in the selection of the whole NE. For example: “希/xi拉/la里/li·克/ke林/lin顿/dun” is a compound name. “希/xi拉/la里/li” can be transliterate to “Hilary” or “Hilaly” and “克/ke林/lin顿/dun” can be transliterate to “Clinton” or “Klinton”. But the combination of “Hilary·Clinton” will be selected for it is the most common combination. So the hit of combination query will be extracted as a feature in classifier.

**Hint words around the NE:** We can take some hint words around the NE into the query, in order to add some limitations to filter out noisy words. For example: “总统 (president)” can be used as hint word for “克林顿 (Clinton)”. To find the hint words, we first search the Chinese name in Chinese web pages. The frequent words can be extracted as hint words and they will be translated to English using a bilingual dictionary. These hint words are combined with the revised candidates to search English web pages. So, the hit of the query will be extracted as feature.

The formula of AdaBoost is as follow.

$$H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right) \quad (7)$$

Where  $\alpha_t$  is the weight for the  $i$ th weak classifier  $h_t(x)$ .  $\alpha_t$  can be calculated based on the precision of its corresponding classifier.

## 5 Experiments

We carry out experiments to investigate how much the revision process and the re-ranking process can improve the performance compared with the baseline of statistical transliteration model. We will also evaluate to which extents we can solve the two problems mentioned in section 1 with the assistance of Web resources.

### 5.1 Experimental data

The training corpus for statistical transliteration model comes from the corpus of Chinese <-> English Name Entity Lists v 1.0 (LDC2005T34). It contains 565,935 transliteration pairs. Ruling out those pairs which are not suitable for the research on Chinese-English backward transliteration, such as Chinese-Japanese, we select a training set which contains 14,443 pairs of Chinese-European & American person names. In the training set, 1,344

pairs are selected randomly as the close test data. 1,294 pairs out of training set are selected as the open test data. To set up the word list, a 2GB-sized collection of web pages is used. Since 7.42% of the names in the test data don’t appear in the list, we use *Google* to get the web page containing the absent names and add these pages into the collection. The word list contains 672,533 words.

### 5.2 Revision phase vs. statistical approach

Using the results generated from statistical model as baseline, we evaluate the revision module in recall first. The statistical transliteration model works in the following 4 steps: 1) Chinese name are transformed into pinyin representation and the English names are split into syllables. 2) The *GIZA++*<sup>1</sup> tool is invoked to align pinyin to syllables, and the alignment probabilities  $P(py|es)$  are obtained. 3) Those frequent sequences of syllables are combined as phrases. For example, “be/r/g” $\rightarrow$ ”berg”, “s/ky” $\rightarrow$ ”sky”. 4) *Camel*<sup>2</sup> decoder is executed to generate 100-best candidates for every name.

We compare the statistical transliteration results with the revised results in Table 2. From Table 2 we can find that the recall of top-100 after revision is improved by 13.26% in close test set and 17.55% in open test set. It proves that the revision module is effective for correcting the mistakes made in statistical transliteration model.

	Transliteration results		Revised results	
	close	open	close	open
Top1	33.64%	9.41%	27.15%	11.04%
Top5	40.37%	13.38%	42.83%	19.69%
Top10	47.79%	17.56%	56.98%	26.52%
Top20	61.88%	25.44%	71.05%	37.81%
Top50	66.49%	36.19%	82.16%	46.22%
Top100	<b>72.52%</b>	<b>41.73%</b>	<b>85.78%</b>	<b>59.28%</b>

Table 2. Statistical model vs. Revision module

To show the effects of the revision on the two above-mentioned problems in which the statistical model does not solve well: the losing of silent syllables and the selection bias problem, we make a statistics of the improvements with a measurement of “correction time”.

For a Chinese word whose correct transliteration appears in top-100 candidates only if it has been

<sup>1</sup> <http://www.fjoch.com/GIZA++.html>

<sup>2</sup> <http://www.nlp.org.cn>

revised, we count the “correction time”. For example, when “Argahi” is revised to “Agassi” the correction time is “1” for Problem II and “1” for Problem I, because in “hi” $\rightarrow$ “si” the syllable is expanded, and in “si” $\rightarrow$ “ssi” an “s” is added.

	Close test	Open test
Problem I	0.6931	0.7853
Problem II	0.9264	1.1672

Table 3. Average time of correction

This measurement reflects the efficiency of the revision of search strategy, in contrast to those spelling correction techniques in which several operations of “add” and “expand” are inevitable. It has proved that the more an average correction time is, the more efficient our strategy is.

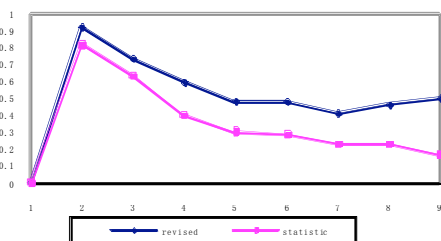


Figure 2. Length influence in recall comparison

The recall of the statistical model relies on the length of English name in some degree. It is more difficult to obtain an absolutely correct answer for longer names, because they may contain more silent and confused syllables. However, through the revision phase, this tendency can be effectively alleviated. In Figure 2, we make a comparison between the results of the statistical model and the revision module with the changing of syllable’s length in open test. The curves demonstrate that the revision indeed prevents the decrease of recall for longer names.

### 5.3 Parameter setting in the revision phase

We will show the experimental results when setting different parameters for query expansion. In the expansion based on phonetic similarity, for every Chinese pinyin, we select at most 20 syllables to create an expansion set. We set  $\beta = 0.1$  in formula (5). The results are shown in the columns labeled “exp1” in Table 4.

From the results we can conclude that, we get the best performance when  $\alpha = 0.4$ . That means the performance is best when the weight of exact

matching is a little larger than the weight of fuzzy matching. We can also see that, higher weight of exact matching will lead to low recall, while higher weight of fuzzy matching will bring noise in.

The expansion method based on syllable similarity is also evaluated. For every syllable, we select at most 15 syllables to create the expansion set. We set  $\beta = 0.1$ . The results are shown in the columns labeled “exp2” in Table 4.

From the results we can conclude that, we get the best performance when  $\alpha = 0.5$ . It means that we can’t put emphasis on any matching methods. Comparison with the expansion based on phonetic similarity, the performance is poorer. It means that the expansion based on phonetic similarity is more suitable for revising transliteration candidates.

### 5.4 Revision phase vs. re-ranking phase

After the phase of revising transliteration candidates, we re-rank the revised candidate set with the assistance of monolingual web resources. In this section, we will show the improvement in precision after re-ranking.

We have selected four kinds of features to integrate in the AdaBoost framework. To determine whether the candidate is NE or not in its context, we use the software tool *Lingpipe*<sup>3</sup>. The queries are sent to *google*, so that we can get the hit of queries and the top-10 snippets will be extracted as context.

The comparison of revision results and re-ranking results is shown as follows.

	Revised results		Re-ranked results	
	close	open	close	open
Top1	27.15%	11.04%	<b>58.08%</b>	<b>38.63%</b>
Top5	42.83%	19.69%	<b>76.35%</b>	<b>52.19%</b>
Top10	56.98%	26.52%	83.92%	54.33%
Top20	71.05%	37.81%	83.92%	57.61%
Top50	82.16%	46.22%	83.92%	57.61%
Top100	<b>85.78%</b>	<b>59.28%</b>	85.78%	59.28%

Table 5. Revision results vs. Re-ranking results

From these results we can conclude that, after re-ranking phase, the noisy words will get a lower

<sup>3</sup> <http://www.alias-i.com/lingpipe/>

	$\alpha = 0.2$		$\alpha = 0.3$		$\alpha = 0.4$		$\alpha = 0.5$		$\alpha = 0.6$		$\alpha = 0.7$		$\alpha = 0.8$	
	exp1	exp2	exp1	exp2	exp1	exp2	exp1	exp2	exp1	exp2	exp1	exp2	exp1	exp2
Top1	13.46	13.32	13.79	13.61	<b>11.04</b>	12.70	11.65	<b>10.93</b>	10.83	11.25	9.62	10.63	8.73	10.18
Top5	21.58	19.59	23.27	20.17	<b>19.69</b>	18.28	21.07	<b>17.25</b>	22.05	16.84	17.90	16.26	17.38	15.34
Top10	27.39	22.71	28.41	24.73	<b>26.52</b>	22.93	26.83	<b>21.81</b>	27.26	20.39	24.38	21.20	25.42	18.20
Top20	35.23	34.88	35.94	29.49	<b>37.81</b>	31.57	38.59	<b>33.04</b>	36.52	31.72	35.25	29.75	34.65	27.62
Top50	43.91	40.63	43.75	40.85	<b>46.22</b>	41.46	48.72	<b>42.79</b>	45.48	40.49	41.57	39.94	42.81	38.07
Top100	53.76	48.47	54.38	52.04	<b>59.28</b>	53.15	57.36	<b>53.46</b>	55.19	51.83	55.63	49.52	53.41	47.15

Table 4. Parameters Experiment

rank. Through the revision module, we get both higher recall and higher precision than statistical transliteration model when at most 5 results are returned.

We also use the average rank and average reciprocal rank (ARR) [Voorhees and Tice, 2000] to evaluate the improvement. ARR is calculated as

$$ARR = \frac{1}{M} \sum_{i=1}^M \frac{1}{R(i)} \quad (8)$$

where  $R(i)$  is the rank of the answer of  $i$ th test word.  $M$  is the size of test set. The higher of ARR, the better the performance is.

The results are shown as Table 6.

	Statistical model		Revision module		Re-rank Module	
	close	open	close	open	close	open
Average rank	37.63	70.94	24.52	58.09	16.71	43.87
ARR	0.3815	0.1206	0.3783	0.1648	0.6519	0.4492

Table 6. ARR and AR evaluation

The ARR after revision phase is lower than the statistical model. Because the goal of revision module is to improve the recall as possible as we can, some noisy words will be introduced in. The noisy words will be pruned in re-ranking module. That is why we get the highest ARR value at last. So we can conclude that the revision module improves recall and re-ranking module improves precision, which help us get a better performance than pure statistical transliteration model

## 6 Conclusion

In this paper, we present a new approach which can revise the results generated from statistical transliteration model with the assistance of monolingual web resource. Through the revision process, the recall of transliteration results has been improved from 72.52% to 85.78% in the close test set and from 41.73% to 59.28% in open test set, respectively. We improve the precision in re-ranking phase, the top-5 precision can be improved to 76.35% in close test and 52.19% in open test. The

promising results show that our approach works pretty well in the task of backward transliteration.

In the future, we will try to improve the similarity measurement in the revision phase. And we also wish to develop a new approach using the transliteration candidates to search for their right answer more directly and effectively.

## Acknowledgments

The work is supported by the National High Technology Development 863 Program of China under Grants no. 2006AA01Z144, the National Natural Science Foundation of China under Grants No. 60673042, the Natural Science Foundation of Beijing under Grants no. 4073043.

## References

- Yaser Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In Proc.of ACL-02.
- Kevin Knight and Jonathan Graehl. 1998. Machine Transliteration. Computational Linguistics 24(4).
- Wei-Hao Lin and Hsin-His Chen. 2002. Backward Machine Transliteration by Learning Phonetic Similarity. In Proc. Of the 6th CoNLL
- Donghui Feng, Yajuan Lv, and Ming Zhou. 2004. A New Approach for English-Chinese Named Entity Alignment. In Proc. of EMNLP-2004.
- Long Jiang, Ming Zhou, Lee-Feng Chien, and Cheng Niu, 2007. Named Entity Translation with Web Mining and Transliteration. In Proc. of IJCAI-2007.
- Wei Gao. 2004. Phoneme-based Statistical Transliteration of Foreign Name for OOV Problem. A thesis of Master. The Chinese University of Hong Kong.
- Ying Zhang, Fei Huang, Stephan Vogel. 2005. Mining translations of OOV terms from the web through cross-lingual query expansion. SIGIR 2005.
- Pu-Jen Cheng, Wen-Hsiang Lu, Jer-Wen Teng, and Lee-Feng Chien. 2004. Creating Multilingual Translation Lexicons with Regional Variations Using Web Corpora. In Proc. of ACL-04
- Masaaki Nagata, Teruka Saito, and Kenji Suzuki. 2001. Using the Web as a Bilingual Dictionary. In Proc. of ACL 2001 Workshop on Data-driven Methods in Machine Translation.



- Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In Proc. of the ACL workshop on Multilingual Named Entity Recognition.
- Jenq-Haur Wang, Jei-Wen Teng, Pu-Jen Cheng, Wen-Hsiang Lu, Lee-Feng Chien. 2004. Translating unknown cross-lingual queries in digital libraries using a web-based approach. In Proc. of JCDL 2004.
- E.M.Voorhees and D.M.Tice. 2000. The trec-8 question answering track report. In Eighth Text Retrieval Conference (TREC-8)

# Robustness and Generalization of Role Sets: PropBank vs. VerbNet

**Beñat Zapirain and Eneko Agirre**  
IXA NLP Group  
University of the Basque Country  
{benat.zapirain,e.agirre}@ehu.es

**Lluís Màrquez**  
TALP Research Center  
Technical University of Catalonia  
lluism@lsi.upc.edu

## Abstract

This paper presents an empirical study on the robustness and generalization of two alternative role sets for semantic role labeling: PropBank numbered roles and VerbNet thematic roles. By testing a state-of-the-art SRL system with the two alternative role annotations, we show that the PropBank role set is more robust to the lack of verb-specific semantic information and generalizes better to infrequent and unseen predicates. Keeping in mind that thematic roles are better for application needs, we also tested the best way to generate VerbNet annotation. We conclude that tagging first PropBank roles and mapping into VerbNet roles is as effective as training and tagging directly on VerbNet, and more robust for domain shifts.

## 1 Introduction

Semantic Role Labeling is the problem of analyzing clause predicates in open text by identifying arguments and tagging them with semantic labels indicating the role they play with respect to the verb. Such sentence-level semantic analysis allows to determine “who” did “what” to “whom”, “when” and “where”, and, thus, characterize the participants and properties of the *events* established by the predicates. This kind of semantic analysis is very interesting for a broad spectrum of NLP applications (information extraction, summarization, question answering, machine translation, etc.), since it opens the door to exploit the semantic relations among linguistic constituents.

The properties of the semantically annotated corpora available have conditioned the type of research

and systems that have been developed so far. PropBank (Palmer et al., 2005) is the most widely used corpus for training SRL systems, probably because it contains running text from the Penn Treebank corpus with annotations on all verbal predicates. Also, a few evaluation exercises on SRL have been conducted on this corpus in the CoNLL-2004 and 2005 conferences. However, a serious criticism to the PropBank corpus refers to the role set it uses, which consists of a set of numbered core arguments, whose semantic translation is verb-dependent. While Arg0 and Arg1 are intended to indicate the general roles of Agent and Theme, other argument numbers do not generalize across verbs and do not correspond to general semantic roles. This fact might compromise generalization and portability of SRL systems, especially when the training corpus is small.

More recently, a mapping from PropBank numbered arguments into VerbNet thematic roles has been developed and a version of the PropBank corpus with thematic roles has been released (Loper et al., 2007). Thematic roles represent a compact set of verb-independent general roles widely used in linguistic theory (e.g., Agent, Theme, Patient, Recipient, Cause, etc.). We foresee two advantages of using such thematic roles. On the one hand, statistical SRL systems trained from them could generalize better and, therefore, be more robust and portable, as suggested in (Yi et al., 2007). On the other hand, roles in a paradigm like VerbNet would allow for inferences over the assigned roles, which is only possible in a more limited way with PropBank.

In a previous paper (Zapirain et al., 2008), we presented a first comparison between the two previous role sets on the SemEval-2007 Task 17 corpus (Pradhan et al., 2007). The SemEval-2007 corpus only

comprised examples about 50 different verbs. The results of that paper were, thus, considered preliminary, as they could depend on the small amount of data (both in training data and number of verbs) or the specific set of verbs being used. Now, we extend those experiments to the entire PropBank corpus, and we include two extra experiments on domain shifts (using the Brown corpus as test set) and on grouping VerbNet labels. More concretely, this paper explores two aspects of the problem. First, having in mind the claim that general thematic roles should be more robust to changing domains and unseen predicates, we study the performance of a state-of-the-art SRL system trained on either codification of roles and some specific settings, i.e. including/excluding verb-specific information, labeling unseen verb predicates, or domain shifts. Second, assuming that application scenarios would prefer dealing with general thematic role labels, we explore the best way to label a text with thematic roles, namely, by training directly on VerbNet roles or by using the PropBank SRL system and perform a posterior mapping into thematic roles.

The results confirm our preliminary findings (Zapirain et al., 2008). We observe that the PropBank roles are more robust in all tested experimental conditions, i.e., the performance decrease is more severe for VerbNet. Besides, tagging first PropBank roles and then mapping into VerbNet roles is as effective as training and tagging directly on VerbNet, and more robust for domain shifts.

The rest of the paper is organized as follows: Section 2 contains some background on PropBank and VerbNet role sets. Section 3 presents the experimental setting and the base SRL system used for the role set comparisons. In Section 4 the main comparative experiments on robustness are described. Section 5 is devoted to analyze the posterior mapping of PropBank outputs into VerbNet thematic roles, and includes results on domain-shift experiments using Brown as test set. Finally, Sections 6 and 7 contain a discussion of the results.

## 2 Corpora and Semantic Role Sets

The PropBank corpus is the result of adding a semantic layer to the syntactic structures of Penn Treebank II (Palmer et al., 2005). Specifically, it pro-

vides information about predicate-argument structures to all verbal predicates of the Wall Street Journal section of the treebank. The role set is theory-neutral and consists of a set of numbered core arguments (Arg0, Arg1, ..., Arg5). Each verb has a *frameset* listing its allowed role labels and mapping each numbered role to an English-language description of its semantics.

Different senses for a polysemous verb have different framesets, but the argument labels are semantically consistent in all syntactic alternations of the same verb-sense. For instance in “Kevin broke [the window]<sub>Arg1</sub>” and in “[The door]<sub>Arg1</sub> broke into a million pieces”, for the verb *broke.01*, both Arg1 arguments have the same semantic meaning, that is “broken entity”. Nevertheless, argument labels are not necessarily consistent across different verbs (or verb senses). For instance, the same Arg2 label is used to identify the Destination argument of a proposition governed by the verb *send* and the Beneficiary argument of the verb *compose*. This fact might compromise generalization of systems trained on PropBank, which might be focusing too much on verb-specific knowledge. It is worth noting that the two most frequent arguments, Arg0 and Arg1, are intended to indicate the general roles of Agent and Theme and are usually consistent across different verbs. However, this correspondence is not total. According to the study by (Yi et al., 2007), Arg0 corresponds to Agent 85.4% of the time, but also to Experiencer (7.2%), Theme (2.1%), and Cause (1.9%). Similarly, Arg1 corresponds to Theme in 47.0% of the occurrences but also to Topic (23.0%), Patient (10.8%), and Product (2.9%), among others. Contrary to core arguments, adjuncts (Temporal and Location markers, etc.) are annotated with a closed set of general and verb-independent labels.

VerbNet (Kipper et al., 2000) is a computational verb lexicon in which verbs are organized hierarchically into classes depending on their syntactic/semantic linking behavior. The classes are based on Levin’s verb classes (Levin, 1993) and each contains a list of member verbs and a correspondence between the shared syntactic frames and the semantic information, such as thematic roles and selectional constraints. There are 23 thematic roles (Agent, Patient, Theme, Experiencer, Source, Beneficiary, Instrument, etc.) which, unlike the Prop-

Bank numbered arguments, are considered as general verb-independent roles.

This level of abstraction makes them, in principle, better suited (compared to PropBank numbered arguments) for being directly exploited by general NLP applications. But, VerbNet by itself is not an appropriate resource to train SRL systems. As opposed to PropBank, the number of tagged examples is far more limited in VerbNet. Fortunately, in the last years a twofold effort has been made in order to generate a large corpus fully annotated with thematic roles. Firstly, the SemLink<sup>1</sup> resource (Loper et al., 2007) established a mapping between PropBank framesets and VerbNet thematic roles. Secondly, the SemLink mapping was applied to a representative portion of the PropBank corpus and manually disambiguated (Loper et al., 2007). The resulting corpus is currently available for the research community and makes possible comparative studies between role sets.

### 3 Experimental Setting

#### 3.1 Datasets

The data used in this work is the benchmark corpus provided by the SRL shared task of CoNLL-2005 (Carreras and Màrquez, 2005). The dataset, of over 1 million tokens, comprises PropBank sections 02–21 for training, and sections 24 and 23 for development and test, respectively. From the input information, we used part of speech tags and full parse trees (generated using Charniak’s parser) and discarded named entities. Also, we used the publicly available SemLink mapping from PropBank into VerbNet roles (Loper et al., 2007) to generate a replicate of the CoNLL-2005 corpus containing also the VerbNet annotation of roles.

Unfortunately, SemLink version 1.0 does not cover all propositions and arguments in the PropBank corpus. In order to have an homogeneous corpus and not to bias experimental evaluation, we decided to discard all incomplete examples and keep only those propositions that were 100% mapped into VerbNet roles. The resulting corpus contains 56% of the original propositions, that is, over 50,000 propositions in the training set. This subcorpus is much larger than the SemEval-2007 Task 17 dataset used

<sup>1</sup><http://verbs.colorado.edu/semLink/>

in our previous experimental work (Zapirain et al., 2008). The difference is especially noticeable in the diversity of predicates represented. In this case, there are 1,709 different verbs (1,505 lemmas) compared to the 50 verbs of the SemEval corpus. We believe that the size and richness of this corpus is enough to test and extract reliable conclusions on the robustness and generalization across verbs of the role sets under study.

In order to study the behavior of both role sets in out-of-domain data, we made use of the PropBanked Brown corpus (Marcus et al., 1994) for testing, as it is also mapped into VerbNet thematic roles in the SemLink resource. Again, we discarded those propositions that were not entirely mapped into thematic roles (45%).

#### 3.2 SRL System

Our basic Semantic Role Labeling system represents the tagging problem as a Maximum Entropy Markov Model (MEMM). The system uses full syntactic information to select a sequence of constituents from the input text and tags these tokens with Begin/Inside/Outside (BIO) labels, using state-of-the-art classifiers and features. The system achieves very good performance in the CoNLL-2005 shared task dataset and in the SRL subtask of the SemEval-2007 English lexical sample task (Zapirain et al., 2007). Check this paper for a complete description of the system.

When searching for the most likely state sequence, the following constraints are observed<sup>2</sup>:

1. No duplicate argument classes for Arg0–Arg5 PropBank (or VerbNet) roles are allowed.
2. If there is a R-X argument (reference), then there has to be a X argument before (referent).
3. If there is a C-X argument (continuation), then there has to be a X argument before.
4. Before a I-X token, there has to be a B-X or I-X token.
5. Given a predicate, only the arguments described in its PropBank (or VerbNet) lexical entry (i.e., the verbal frameset) are allowed.

<sup>2</sup>Note that some of the constraints are dependent of the role set used, i.e., PropBank or VerbNet

Regarding the last constraint, the lexical entries of the verbs were constructed from the training data itself. For instance, the verb *build* appears with four different PropBank core roles (Arg0–3) and five VerbNet roles (Product, Material, Asset, Attribute, Theme), which are the only ones allowed for that verb at test time. Note that in the cases where the verb sense was known we could constraint the possible arguments to those that appear in the lexical entry of that sense, as opposed of using the arguments that appear in all senses.

#### 4 On the Generalization of Role Sets

We first seek a basic reference of the comparative performance of the classifier on each role set. We devised two settings based on our dataset. In the first setting (‘SemEval’) we use all the available information provided in the corpus, including the verb senses in PropBank and VerbNet. This information was available both in the training and test, and was thus used as an additional feature by the classifier and to constrain further the possible arguments when searching for the most probable Viterbi path. We call this setting ‘SemEval’ because the SemEval-2007 competition (Pradhan et al., 2007) was performed using this configuration.

Being aware that, in a real scenario, the sense information will not be available, we devised the second setting (‘CoNLL’), where the hand-annotated verb sense information was discarded. This is the setting used in the CoNLL 2005 shared task (Carreras and Màrquez, 2005).

The results for the first setting are shown in the ‘SemEval setting’ rows of Table 1. The correct, excess, missed, precision, recall and  $F_1$  measures are reported, as customary. The significance intervals for  $F_1$  are also reported. They have been obtained with bootstrap resampling (Noreen, 1989).  $F_1$  scores outside of these intervals are assumed to be significantly different from the related  $F_1$  score ( $p < 0.05$ ). The results for PropBank are slightly better, which is reasonable, as the number of labels that the classifier has to learn in the case of VerbNet should make the task harder. In fact, given the small difference, one could think that VerbNet labels, being more numerous, are easier to learn, perhaps because they are more consistent across verbs.

In the second setting (‘CoNLL setting’ row in the same table) the PropBank classifier degrades slightly, but the difference is not statistically significant. On the contrary, the drop of 1.6 points for VerbNet is significant, and shows greater sensitivity to the absence of the sense information for verbs. One possible reason could be that the VerbNet classifier is more dependant on the argument filter (i.e., the 5th constraint in Section 3.2, which only allows roles that occur in the verbal frameset) used in the Viterbi search, and lacking the sense information makes the filter less useful. In fact, we have attested that the 5th constrain discard more than 60% of the possible candidates for VerbNet, making the task of the classifier easier.

In order to test this hypothesis, we run the CoNLL setting with the 5th constraint disabled (that is, allowing any argument). The results in the ‘CoNLL setting (no 5th)’ rows of Table 1 show that the drop for PropBank is negligible and not significant, while the drop for VerbNet is more important, and statistically significant.

Another view of the data is obtained if we compute the  $F_1$  scores for core arguments and adjuncts separately (last two columns in Table 1). The performance drop for PropBank in the first three rows is equally distributed on both core arguments and adjuncts. On the contrary, the drop for VerbNet roles is more acute in core arguments (3.7 points), while adjuncts with the 5th constraint disabled get results close to the SemEval setting. These results confirm that the information in the verbal frameset is more important in VerbNet than in PropBank, as only core arguments are constrained in the verbal framesets. The explanation could stem from the fact that current SRL systems rely more on syntactic information than pure semantic knowledge. While PropBank arguments Arg0–5 are easier to distinguish on syntactic grounds alone, it seems quite difficult to distinguish among roles like Theme and Topic unless we have access to the specific verbal frameset. This corresponds nicely with the performance drop for VerbNet when there is less information about the verb in the algorithm (i.e., sense or frameset).

We further analyzed the results by looking at each of the individual core arguments and adjuncts. Table 2 shows these results on the CoNLL setting. The performance for the most frequent roles is similar

PropBank								
Experiment	correct	excess	missed	precision	recall	F <sub>1</sub>	F <sub>1</sub> core	F <sub>1</sub> adj.
SemEval setting	6,022	1,378	1,722	81.38	77.76	79.53 ±0.9	82.25	72.48
CoNLL setting	5,977	1,424	1,767	80.76	77.18	78.93 ±0.9	81.64	71.90
CoNLL setting (no 5th)	5,972	1,434	1,772	80.64	77.12	78.84 ±0.9	81.49	71.50
No verbal features	5,557	1,828	2,187	75.25	71.76	73.46 ±1.0	74.87	70.11
Unseen verbs	267	89	106	75.00	71.58	73.25 ±4.0	76.21	64.92

VerbNet								
Experiment	correct	excess	missed	precision	recall	F <sub>1</sub>	F <sub>1</sub> core	F <sub>1</sub> adj.
SemEval setting	5,927	1,409	1,817	80.79	76.54	78.61 ±0.9	81.28	71.83
CoNLL setting	5,816	1,548	1,928	78.98	75.10	76.99 ±0.9	79.44	70.20
CoNLL setting (no 5th)	5,746	1,669	1,998	77.49	74.20	75.81 ±0.9	77.60	71.67
No verbal features	4,679	2,724	3,065	63.20	60.42	61.78 ±0.9	59.19	69.95
Unseen verbs	207	136	166	60.35	55.50	57.82 ±4.3	55.04	63.41

Table 1: Basic results using PropBank (top) and VerbNet (bottom) role sets on different settings.

for both. Arg0 gets 88.49, while Agent and Experiencer get 87.31 and 87.76 respectively. Arg2 gets 79.91, but there is more variation on Theme, Topic and Patient (which get 75.46, 85.70 and 78.64 respectively).

Finally, we grouped the results according to the frequency of the verbs in the training data. Table 3 shows that both PropBank and VerbNet get decreasing results for less frequent verbs. PropBank gets better results in all frequency ranges, except for the most frequent, which contains a single verb (*say*).

Overall, the results on this section point out at the weaknesses of the VerbNet role set regarding robustness and generalization. The next sections examine further its behavior.

#### 4.1 Generalization to Unseen Predicates

In principle, the PropBank core roles (Arg0–4) get a different interpretation depending of the verb, that is, the meaning of each of the roles is described separately for each verb in the PropBank framesets. Still, the annotation criteria used with PropBank tried to make the two main roles (Arg0 and Arg1, which account for most of the occurrences) consistent across verbs. On the contrary, in VerbNet all roles are completely independent of the verb, in the sense that the interpretation of the role does not vary across verbs. But, at the same time, each verbal entry lists the possible roles it accepts, and the combinations allowed.

This experiment tests the sensitivity of the two approaches when the SRL system encounters a verb which does not occur in the training data. In principle, we would expect the VerbNet semantic labels, which are more independent across verbs, to be

more robust at tagging new predicates. It is worth noting that this is a realistic scenario, even for the verb-specific PropBank labels. Predicates which do not occur in the training data, but do have a PropBank lexicon entry, could appear quite often in the text to be analyzed.

For this experiment, we artificially created a test set for unseen verbs. We chose 50 verbs at random, and split them into 40 verbs for training and 10 for testing (yielding 13,146 occurrences for training and 2,723 occurrences for testing; see Table 4).

The results obtained after training and testing the classifier are shown in the last rows in Table 1. Note that they are not directly comparable to the other results mentioned so far, as the train and test sets are smaller. Figures indicate that the performance of the PropBank argument classifier is considerably higher than the VerbNet classifier, with a  $\sim 15$  point gap.

This experiment shows that lacking any information about verbal head, the classifier has a hard time to distinguish among VerbNet roles. In order to confirm this, we performed the following experiment.

#### 4.2 Sensitivity to Verb-dependent Features

In this experiment we want to test the sensitivity of the role sets when the classifier does not have any information of the verb predicate. We removed from the training and testing data all the features which make any reference to the verb, including, among others: the surface form, lemma and POS of the verb, and all the combined features that include the verb form (please, refer to (Zapirain et al., 2007) for a complete description of the feature set).

The results are shown in the ‘No verbal features’

	CoNLL setting				No verb features	
	PBank		VNet		PBank	VNet
	corr.	F <sub>1</sub>	corr.	F <sub>1</sub>	F <sub>1</sub>	F <sub>1</sub>
Overall	5977	78.93	5816	76.99	73.46	61.78
Arg0	1919	88.49			84.02	
Arg1	2240	79.81			73.29	
Arg2	303	65.44			48.58	
Arg3	10	52.63			14.29	
Actor1			44	85.44		0.00
Actor2			10	71.43		25.00
Agent			1603	87.31		77.21
Attribut.			25	71.43		50.79
Cause			51	62.20		5.61
Experien.			215	87.76		86.69
Location			31	64.58		25.00
Patient1			38	67.86		5.71
Patient			208	78.64		25.06
Patient2			21	67.74		43.33
Predicate			83	62.88		28.69
Product			44	61.97		2.44
Recipient			85	79.81		62.73
Source			29	60.42		30.95
Stimulus			39	63.93		13.70
Theme			1021	75.46		52.14
Theme1			20	57.14		4.44
Theme2			21	70.00		23.53
Topic			683	85.70		73.58
ADV	132	53.44	129	52.12	52.67	53.31
CAU	13	53.06	13	52.00	53.06	45.83
DIR	22	53.01	27	56.84	40.00	46.34
DIS	133	77.78	137	79.42	77.25	78.34
LOC	126	61.76	126	61.02	59.56	57.34
MNR	109	58.29	111	54.81	52.99	51.49
MOD	249	96.14	248	95.75	96.12	95.57
NEG	124	98.41	124	98.80	98.41	98.01
PNC	26	44.07	29	44.62	38.33	41.79
TMP	453	75.00	450	73.71	73.06	73.89

Table 2: Detailed results on the CoNLL setting. Reference arguments and verbs have been omitted for brevity, as well as those with less than 10 occ. The last two columns refer to the results on the CoNLL setting with no verb features.

Freq.	PBank	VNet	Freq.	PBank	VNet
0-50	74,21	71,11	500-900	77,97	75,77
50-100	74,79	71,83	> 900	91,83	92,23
100-500	77,16	75,41			

Table 3: F<sub>1</sub> results split according to the frequency of the verb in the training data.

Train	<i>affect, announce, ask, attempt, avoid, believe, build, care, cause, claim, complain, complete, contribute, describe, disclose, enjoy, estimate, examine, exist, explain, express, feel, fix, grant, hope, join, maintain, negotiate, occur, prepare, promise, propose, purchase, recall, receive, regard, remember, remove, replace, say</i>
Test	<i>allow, approve, buy, find, improve, kill, produce, prove, report, rush</i>

Table 4: Verbs used in the *unseen verb* experiment

rows of Table 1. The performance drops more than 5 points in PropBank, but the drop for VerbNet is dramatic, with more than 15 points.

A closer look at the detailed role-by-role performances can be done if we compare the F<sub>1</sub> rows in the CoNLL setting and in the ‘no verb features’ setting in Table 2. Those results show that both Arg0 and Arg1 are quite robust to the lack of target verb information, while Arg2 and Arg3 get more affected. Given the relatively low number of Arg2 and Arg3 arguments, their performance drop does not affect so much the overall PropBank performance. In the case of VerbNet, the picture is very different. Focusing on the most frequent roles first, while the performance drop for Experiencer, Agent and Topic is of 1, 10 and 12 points respectively, the other roles get very heavy losses (e.g. Theme and Patient drop 23 and 50 points), and the rest of roles are barely found. It is worth noting that the adjunct labels get very similar performances in both PropBank and VerbNet cases. In fact, Table 1 in the last two rows shows very clearly that the performance drop is caused by the core arguments.

The better robustness of the PropBank roles can be explained by the fact that, when creating PropBank, the human PropBank annotators tried to be consistent when tagging Arg0 and Arg1 across verbs. We also think that both Arg0 and Arg1 can be detected quite well relying on unlexicalized syntactic features only, that is, not knowing which are the verbal and nominal heads. On the other hand, distinguishing between Arg2–4 is more dependant on the subcategorization frame of the verb, and thus more sensitive to the lack of verbal information.

In the case of VerbNet, the more fine-grained distinction among roles seems to depend more on the meaning of the predicate. For instance, distinguishing between Agent–Experiencer, or Theme–Topic–Patient. The lack of the verbal head makes it much more difficult to distinguish among those roles. The same phenomena can be observed among the roles not typically realized as Subject or Object such as Recipient, Source, Product, or Stimulus.

## 5 Mapping into VerbNet Thematic Roles

As mentioned in the introduction, the interpretation of PropBank roles depends on the verb, and that

Test on WSJ	all	core	adj.
PropBank to VerbNet (hand)	79.17 ±0.9	81.77	72.50
VerbNet (SemEval setting)	78.61 ±0.9	81.28	71.84
PropBank to VerbNet (MF)	77.15 ±0.9	79.09	71.90
VerbNet (CoNLL setting)	76.99 ±0.9	79.44	70.88
Test on Brown			
PropBank to VerbNet (MF)	64.79 ±1.0	68.93	55.94
VerbNet (CoNLL setting)	62.87 ±1.0	67.07	54.69

Table 5: Results on VerbNet roles using two different strategies. Topmost 4 rows for the usual test set (WSJ), and the 2 rows below for the Brown test set.

makes them less suitable for NLP applications. On the other hand, VerbNet roles have a direct interpretation. In this section, we test the performance of two different approaches to tag input sentences with VerbNet roles: (1) train on corpora tagged with VerbNet, and tag the input directly; (2) train on corpora tagged with PropBank, tag the input with PropBank roles, and use a PropBank to VerbNet mapping to output VerbNet roles.

The results for the first approach are already available (cf. Table 1). For the second approach, we just need to map PropBank roles into VerbNet roles using SemLink (Loper et al., 2007). We devised two experiments. In the first one we use the hand-annotated verb class in the test set. For each predicate we translate PropBank roles into VerbNet roles making use of the SemLink mapping information corresponding to that verb lemma and its verbal class.

For instance, consider an occurrence of *allow* in a test sentence. If the occurrence has been manually annotated with the VerbNet class 29.5, we can use the following entry in SemLink to add the VerbNet role Predicate to the argument labeled with Arg1, and Agent to the Arg0 argument.

```
<predicate lemma="allow">
  <argmap pb-roleset="allow.01" vn-class="29.5">
    <role pb-arg="1" vn-theta="Predicate" />
    <role pb-arg="0" vn-theta="Agent" />
  </argmap>
</predicate>
```

The results obtained using the hand-annotated VerbNet classes (and the SemEval setting for PropBank), are shown in the first row of Table 5. If we compare these results to those obtained by VerbNet in the SemEval setting (second row of Table 5), they are 0.5 points better, but the difference is not statistically significant.

experiment	corr.	F <sub>1</sub>
Grouped (CoNLL Setting)	5,951	78.11±0.9
PropBank to VerbNet to Grouped	5,970	78.21±0.9

Table 6: Results for VerbNet grouping experiments.

In a second experiment, we discarded the sense annotations from the dataset, and tried to predict the VerbNet class of the target verb using the most frequent class for the verb in the training data. Surprisingly, the accuracy of choosing the most frequent class is 97%. In the case of *allow* the most frequent class is 29.5, so we would use the same SemLink entry as above. The third row in Table 5 shows the results using the most frequent VerbNet class (and the CoNLL setting for PropBank). The performance drop compared to the use of the hand-annotated VerbNet class is of 2 points and statistically significant, and 0.2 points above the results obtained using VerbNet directly on the same conditions (fourth row of the same Table).

The last two rows in table 5 show the results when testing on the the Brown Corpus. In this case, the difference is larger, 1.9 points, and statistically significant in favor of the mapping approach. These results show that VerbNet roles are less robust to domain shifts. The performance drop when moving to an out-of-domain corpus is consistent with previously published results (Carreras and Màrquez, 2005).

## 5.1 Grouping experiments

VerbNet roles are more numerous than PropBank roles, and that, in itself, could cause a drop in performance. Motivated by the results in (Yi et al., 2007), we grouped the 23 VerbNet roles in 7 coarser role groups. Note that their groupings are focused on the roles which map to PropBank Arg2. In our case we are interested in a more general grouping which covers all VerbNet roles, so we added two additional groups (Agent-Experiencer and Theme-Topic-Patient). We re-tagged the roles in the datasets with those groups, and then trained and tested our SRL system on those grouped labels. The results are shown in the first row of Table 6. In order to judge if our groupings are easier to learn, we can see that the performance gain with respect to the ungrouped roles (fourth row of Table 5) is small (76.99



vs. 78.11) but significant. But if we compare them to the results of the PropBank to VerbNet mapping, where we simply substitute the fine-grained roles by their corresponding groups, we see that they still lag behind (second row in Table 6).

Although one could argue that better motivated groupings could be proposed, these results indicate that the larger number of VerbNet roles does not explain in itself the performance difference when compared to PropBank.

## 6 Related Work

As far as we know, there are only two other works performing comparisons of alternative role sets on a common test data. Gildea and Jurafsky (2002) mapped FrameNet frame elements into a set of *abstract thematic roles* (i.e., more general roles such as Agent, Theme, Location), and concluded that their system could use these thematic roles without degradation in performance.

(Yi et al., 2007) is a closely related work. They also compare PropBank and VerbNet role sets, but they focus on the performance of Arg2. They show that splitting Arg2 instances into subgroups based on VerbNet thematic roles improves the performance of the PropBank-based classifier. Their claim is that since VerbNet uses argument labels that are more consistent across verbs, they would provide more consistent training instances which would generalize better, especially to new verbs and genres. In fact they get small improvements in PropBank (WSJ) and a large improvement when testing on Brown.

An important remark is that Yi et al. use a combination of grouped VerbNet roles (for Arg2) and PropBank roles (for the rest of arguments). In contrast, our study compares both role sets as they stand, without modifications or mixing. Another difference is that they compare the systems based on the PropBank roles —by mapping the output VerbNet labels back to PropBank Arg2— while in our case we decided to do just the contrary (i.e., mapping PropBank output into VerbNet labels and compare there). As we already said, we think that VerbNet-based labels can be more useful for NLP applications, so our target is to have a SRL system that provides VerbNet annotations. While not in direct contradiction, both studies show different angles of the complex relation

between the two role sets.

## 7 Conclusion and Future work

In this paper we have presented a study of the performance of a state-of-the-art SRL system trained on two alternative codifications of roles (PropBank and VerbNet) and some particular settings, e.g., including/excluding verb-specific information in features, labeling of infrequent and unseen verb predicates, and domain shifts. We observed that PropBank labeling is more robust in all previous experimental conditions, showing less performance drops than VerbNet labels.

Assuming that application-based scenarios would prefer dealing with general thematic role labels, we explore the best way to label a text with VerbNet thematic roles, namely, by training directly on VerbNet roles or by using the PropBank SRL system and performing a posterior mapping into thematic roles. While results are similar and not statistically significant in the WSJ test set, when testing on the Brown out-of-domain test set the difference in favor of PropBank plus mapping step is statistically significant. We also tried to map the fine-grained VerbNet roles into coarser roles, but it did not yield better results than the mapping from PropBank roles. As a side-product, we show that a simple most frequent sense disambiguation strategy for verbs is sufficient to provide excellent results in the PropBank to VerbNet mapping.

Regarding future work, we would like to explore ways to improve the performance on VerbNet roles, perhaps using selectional preferences. We also want to work on the adaptation to new domains of both roles sets.

## Acknowledgements

We are grateful to Martha Palmer and Edward Loper for kindly providing us with the SemLink mappings. This work has been partially funded by the Basque Government (IT-397-07) and by the Ministry of Education (KNOW TIN2006-15049, OpenMT TIN2006-15307-C03-02). Beñat is supported by a PhD grant from the University of the Basque Country.

## References

- Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In Ido Dagan and Daniel Gildea, editors, *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, USA, June. Association for Computational Linguistics.
- Karin Kipper, Hoa Trang Dang, and Martha Palmer. 2000. Class based construction of a verb lexicon. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX, July.
- Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. The University of Chicago Press, Chicago.
- Edward Loper, Szu-Ting Yi, and Martha Palmer. 2007. Combining lexical resources: Mapping between propbank and verbnet. In *Proceedings of the 7th International Workshop on Computational Linguistics*, Tilburg, the Netherlands.
- Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 114–119, Morristown, NJ, USA. Association for Computational Linguistics.
- Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.
- Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. Semeval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic, June. Association for Computational Linguistics.
- Szu-Ting Yi, Edward Loper, and Martha Palmer. 2007. Can semantic roles generalize across genres? In *Proceedings of the Human Language Technology Conferences/North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT/NAACL-2007)*.
- Beñat Zepirain, Eneko Agirre, and Lluís Màrquez. 2007. Sequential SRL Using Selectional Preferences. An Approach with Maximum Entropy Markov Models. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 354–357.
- Beñat Zepirain, Eneko Agirre, and Lluís Màrquez. 2008. A Preliminary Study on the Robustness and Generalization of Role Sets for Semantic Role Labeling. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing-2008)*, pages 219–230, Haifa, Israel, February.

# A Tree Sequence Alignment-based Tree-to-Tree Translation Model

Min Zhang<sup>1</sup> Hongfei Jiang<sup>2</sup> Aiti Aw<sup>1</sup> Haizhou Li<sup>1</sup> Chew Lim Tan<sup>3</sup> and Sheng Li<sup>2</sup>

<sup>1</sup>Institute for Infocomm Research

mzhang@i2r.a-star.edu.sg

aaiti@i2r.a-star.edu.sg

hli@i2r.a-star.edu.sg

<sup>2</sup>Harbin Institute of Technology

hfjiang@mtlab.hit.edu.cn

lisheng@hit.edu.cn

<sup>3</sup>National University of Singapore

tancl@comp.nus.edu.sg

## Abstract

This paper presents a translation model that is based on tree sequence alignment, where a tree sequence refers to a single sequence of sub-trees that covers a phrase. The model leverages on the strengths of both phrase-based and linguistically syntax-based method. It automatically learns aligned tree sequence pairs with mapping probabilities from word-aligned biparsed parallel texts. Compared with previous models, it not only captures non-syntactic phrases and discontinuous phrases with linguistically structured features, but also supports multi-level structure reordering of tree typology with larger span. This gives our model stronger expressive power than other reported models. Experimental results on the NIST MT-2005 Chinese-English translation task show that our method statistically significantly outperforms the baseline systems.

## 1 Introduction

Phrase-based modeling method (Koehn et al., 2003; Och and Ney, 2004a) is a simple, but powerful mechanism to machine translation since it can model local reorderings and translations of multi-word expressions well. However, it cannot handle long-distance reorderings properly and does not exploit discontinuous phrases and linguistically syntactic structure features (Quirk and Menezes, 2006). Recently, many syntax-based models have been proposed to address the above deficiencies (Wu, 1997; Chiang, 2005; Eisner, 2003; Ding and Palmer, 2005; Quirk et al, 2005; Cowan et al., 2006; Zhang et al., 2007; Bod, 2007; Yamada and Knight, 2001; Liu et al., 2006; Liu et al., 2007; Gildea, 2003; Poutsma, 2000; Hearne and Way,

2003). Although good progress has been reported, the fundamental issues in applying linguistic syntax to SMT, such as non-isomorphic tree alignment, structure reordering and non-syntactic phrase modeling, are still worth well studying.

In this paper, we propose a tree-to-tree translation model that is based on tree sequence alignment. It is designed to combine the strengths of phrase-based and syntax-based methods. The proposed model adopts tree sequence<sup>1</sup> as the basic translation unit and utilizes tree sequence alignments to model the translation process. Therefore, it not only describes non-syntactic phrases with syntactic structure information, but also supports multi-level tree structure reordering in larger span. These give our model much more expressive power and flexibility than those previous models. Experiment results on the NIST MT-2005 Chinese-English translation task show that our method significantly outperforms Moses (Koehn et al., 2007), a state-of-the-art phrase-based SMT system, and other linguistically syntax-based methods, such as SCFG-based and STSG-based methods (Zhang et al., 2007). In addition, our study further demonstrates that 1) structure reordering rules in our model are very useful for performance improvement while discontinuous phrase rules have less contribution and 2) tree sequence rules are able to model non-syntactic phrases with syntactic structure information, and thus contribute much to the performance improvement, but those rules consisting of more than three sub-trees have almost no contribution.

The rest of this paper is organized as follows: Section 2 reviews previous work. Section 3 elabo-

---

<sup>1</sup> A tree sequence refers to an ordered sub-tree sequence that covers a phrase or a consecutive tree fragment in a parse tree. It is the same as the concept “forest” used in Liu et al (2007).

rates the modelling process while Sections 4 and 5 discuss the training and decoding algorithms. The experimental results are reported in Section 6. Finally, we conclude our work in Section 7.

## 2 Related Work

Many techniques on linguistically syntax-based SMT have been proposed in literature. Yamada and Knight (2001) use noisy-channel model to transfer a target parse tree into a source sentence. Eisner (2003) studies how to learn non-isomorphic tree-to-tree/string mappings using a STSG. Ding and Palmer (2005) propose a syntax-based translation model based on a probabilistic synchronous dependency insertion grammar. Quirk et al. (2005) propose a dependency treelet-based translation model. Cowan et al. (2006) propose a feature-based discriminative model for target language syntactic structures prediction, given a source parse tree. Huang et al. (2006) study a TSG-based tree-to-string alignment model. Liu et al. (2006) propose a tree-to-string model. Zhang et al. (2007b) present a STSG-based tree-to-tree translation model. Bod (2007) reports that the unsupervised STSG-based translation model performs much better than the supervised one. The motivation behind all these work is to exploit linguistically syntactic structure features to model the translation process. However, most of them fail to utilize non-syntactic phrases well that are proven useful in the phrase-based methods (Koehn et al., 2003).

The formally syntax-based model for SMT was first advocated by Wu (1997). Xiong et al. (2006) propose a MaxEnt-based reordering model for BTG (Wu, 1997) while Setiawan et al. (2007) propose a function word-based reordering model for BTG. Chiang (2005)'s hierarchal phrase-based model achieves significant performance improvement. However, no further significant improvement is achieved when the model is made sensitive to syntactic structures by adding a constituent feature (Chiang, 2005).

In the last two years, many research efforts were devoted to integrating the strengths of phrase-based and syntax-based methods. In the following, we review four representatives of them.

1) Hassan et al. (2007) integrate supertags (a kind of lexicalized syntactic description) into the target side of translation model and language mod-

el under the phrase-based translation framework, resulting in good performance improvement. However, neither source side syntactic knowledge nor reordering model is further explored.

2) Galley et al. (2006) handle non-syntactic phrasal translations by traversing the tree upwards until a node that subsumes the phrase is reached. This solution requires larger applicability contexts (Marcu et al., 2006). However, phrases are utilized independently in the phrase-based method without depending on any contexts.

3) Addressing the issues in Galley et al. (2006), Marcu et al. (2006) create an xRS rule headed by a pseudo, non-syntactic non-terminal symbol that subsumes the phrase and its corresponding multi-headed syntactic structure; and one sibling xRS rule that explains how the pseudo symbol can be combined with other genuine non-terminals for acquiring the genuine parse trees. The name of the pseudo non-terminal is designed to reflect the full realization of the corresponding rule. The problem in this method is that it neglects alignment consistency in creating sibling rules and the naming mechanism faces challenges in describing more complicated phenomena (Liu et al., 2007).

4) Liu et al. (2006) treat all bilingual phrases as lexicalized tree-to-string rules, including those non-syntactic phrases in training corpus. Although the solution shows effective empirically, it only utilizes the source side syntactic phrases of the input parse tree during decoding. Furthermore, the translation probabilities of the bilingual phrases and other tree-to-string rules are not compatible since they are estimated independently, thus having different parameter spaces. To address the above problems, Liu et al. (2007) propose to use forest-to-string rules to enhance the expressive power of their tree-to-string model. As is inherent in a tree-to-string framework, Liu et al.'s method defines a kind of auxiliary rules to integrate forest-to-string rules into tree-to-string models. One problem of this method is that the auxiliary rules are not described by probabilities since they are constructed during decoding, rather than learned from the training corpus. So, to balance the usage of different kinds of rules, they use a very simple feature counting the number of auxiliary rules used in a derivation for penalizing the use of forest-to-string and auxiliary rules.

In this paper, an alternative solution is presented to combine the strengths of phrase-based and syn-

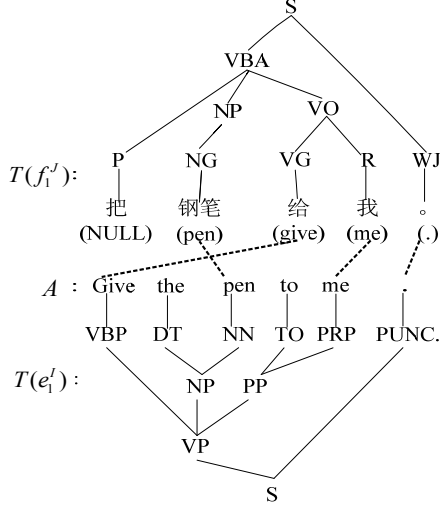


Figure 1: A word-aligned parse tree pairs of a Chinese sentence and its English translation

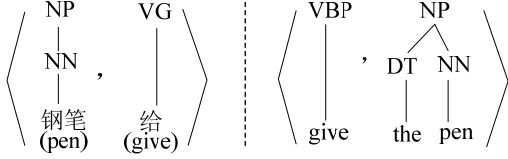


Figure 2: Two Examples of tree sequences

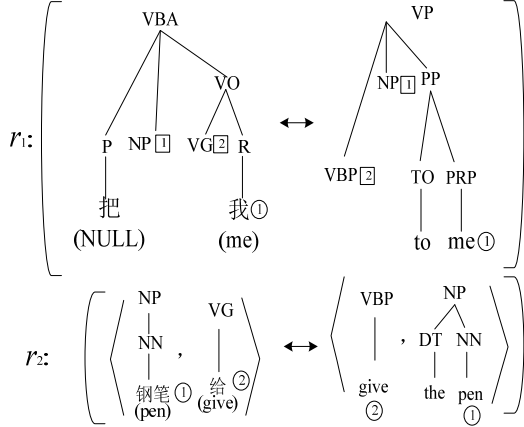


Figure 3: Two examples of translation rules

tax-based methods. Unlike previous work, our solution neither requires larger applicability contexts (Galley et al., 2006), nor depends on pseudo nodes (Marcu et al., 2006) or auxiliary rules (Liu et al., 2007). We go beyond the single sub-tree mapping model to propose a tree sequence alignment-based translation model. To the best of our knowledge, this is the first attempt to empirically explore the tree sequence alignment based model in SMT.

### 3 Tree Sequence Alignment Model

#### 3.1 Tree Sequence Translation Rule

The leaf nodes of a sub-tree in a tree sequence can be either non-terminal symbols (grammar tags) or terminal symbols (lexical words). Given a pair of source and target parse trees  $T(f_1^J)$  and  $T(e_1^I)$  in Fig. 1, Fig. 2 illustrates two examples of tree sequences derived from the two parse trees. A tree sequence translation rule  $r$  is a pair of aligned tree sequences  $r = \langle TS(f_1^{j_2}), TS(e_1^{i_2}), \tilde{A} \rangle$ , where:

- $TS(f_1^{j_2})$  is a source tree sequence, covering the span  $[j_1, j_2]$  in  $T(f_1^J)$ , and
- $TS(e_1^{i_2})$  is a target one, covering the span  $[i_1, i_2]$  in  $T(e_1^I)$ , and
- $\tilde{A}$  are the alignments between leaf nodes of two tree sequences, satisfying the following condition:  $\forall (i, j) \in \tilde{A}: i_1 \leq i \leq i_2 \leftrightarrow j_1 \leq j \leq j_2$ .

Fig. 3 shows two rules extracted from the tree pair shown in Fig. 1, where  $r_1$  is a tree-to-tree rule and  $r_2$  is a tree sequence-to-tree sequence rule. Obviously, tree sequence rules are more powerful than phrases or tree rules as they can capture all phrases (including both syntactic and non-syntactic phrases) with syntactic structure information and allow any tree node operations in a longer span. We expect that these properties can well address the issues of non-isomorphic structure alignments, structure reordering, non-syntactic phrases and discontinuous phrases translations.

#### 3.2 Tree Sequence Translation Model

Given the source and target sentences  $f_1^J$  and  $e_1^I$  and their parse trees  $T(f_1^J)$  and  $T(e_1^I)$ , the tree sequence-to-tree sequence translation model is formulated as:

$$\begin{aligned}
 Pr(e_1^I | f_1^J) &= \sum_{T(f_1^J), T(e_1^I)} Pr(e_1^I, T(e_1^I), T(f_1^J) | f_1^J) \\
 &= \sum_{T(f_1^J), T(e_1^I)} (Pr(T(f_1^J) | f_1^J) \\
 &\quad \cdot Pr(T(e_1^I) | T(f_1^J), f_1^J) \\
 &\quad \cdot Pr(e_1^I | T(e_1^I), T(f_1^J), f_1^J))
 \end{aligned} \tag{1}$$

In our implementation, we have:

- 1)  $Pr(T(f_1^J) | f_1^J) \equiv 1$  since we only use the best source and target parse tree pairs in training.
- 2)  $Pr(e_1^l | T(e_1^l), T(f_1^J), f_1^J) \equiv 1$  since we just output the leaf nodes of  $T(e_1^l)$  to generate  $e_1^l$  regardless of source side information.

Since  $T(f_1^J)$  contains the information of  $f_1^J$ , now we have:

$$\begin{aligned} Pr(e_1^l | f_1^J) &= Pr(T(e_1^l) | T(f_1^J), f_1^J) \\ &= Pr(T(e_1^l) | T(f_1^J)) \end{aligned} \quad (2)$$

By Eq. (2), translation becomes a tree structure mapping issue. We model it using our tree sequence-based translation rules. Given the source parse tree  $T(f_1^J)$ , there are multiple derivations that could lead to the same target tree  $T(e_1^l)$ , the mapping probability  $Pr(T(e_1^l) | T(f_1^J))$  is obtained by summing over the probabilities of all derivations. The probability of each derivation  $\theta$  is given as the product of the probabilities of all the rules  $p(r_i)$  used in the derivation (here we assume that a rule is applied *independently* in a derivation).

$$\begin{aligned} Pr(e_1^l | f_1^J) &= Pr(T(e_1^l) | T(f_1^J)) \\ &= \sum_{\theta} \prod_{r_i \in \theta} p(r_i : \langle TS(e_1^l), TS(f_1^J), \tilde{A} \rangle) \end{aligned} \quad (3)$$

Eq. (3) formulates the tree sequence alignment-based translation model. Figs. 1 and 3 show how the proposed model works. First, the source sentence is parsed into a source parse tree. Next, the source parse tree is detached into two source tree sequences (the left hand side of rules in Fig. 3). Then the two rules in Fig. 3 are used to map the two source tree sequences to two target tree sequences, which are then combined to generate a target parse tree. Finally, a target translation is yielded from the target tree.

Our model is implemented under log-linear framework (Och and Ney, 2002). We use seven basic features that are analogous to the commonly used features in phrase-based systems (Koehn, 2004): 1) bidirectional rule mapping probabilities; 2) bidirectional lexical rule translation probabilities; 3) the target language model; 4) the number of rules used and 5) the number of target words. In addition, we define two new features: 1) the number of lexical words in a rule to control the model's preference for lexicalized rules over un-lexicalized

rules and 2) the average tree depth in a rule to balance the usage of hierarchical rules and flat rules. Note that we do not distinguish between larger (taller) and shorter source side tree sequences, i.e. we let these rules compete directly with each other.

#### 4 Rule Extraction

Rules are extracted from word-aligned, bi-parsed sentence pairs  $\langle T(f_1^J), T(e_1^l), A \rangle$ , which are classified into two categories:

- **initial rule**, if all leaf nodes of the rule are terminals (i.e. lexical word), and
- **abstract rule**, otherwise, i.e. at least one leaf node is a non-terminal (POS or phrase tag).

Given an *initial rule*  $\langle TS(f_{j_1}^{j_2}), TS(e_{i_1}^{i_2}), \tilde{A} \rangle$ , its *sub initial rule* is defined as a triple  $\langle TS(f_{j_3}^{j_4}), TS(e_{i_3}^{i_4}), \hat{A} \rangle$  if and only if:

- $\langle TS(f_{j_3}^{j_4}), TS(e_{i_3}^{i_4}), \hat{A} \rangle$  is an *initial rule*.
- $\forall (i, j) \in \tilde{A} : i_3 \leq i \leq i_4 \leftrightarrow j_3 \leq j \leq j_4$ , i.e.  $\hat{A} \subseteq \tilde{A}$
- $TS(f_{j_3}^{j_4})$  is a sub-graph of  $TS(f_{j_1}^{j_2})$  while  $TS(e_{i_3}^{i_4})$  is a sub-graph of  $TS(e_{i_1}^{i_2})$ .

Rules are extracted in two steps:

- 1) Extracting *initial rules* first.
- 2) Extracting *abstract rules* from extracted *initial rules* with the help of *sub initial rules*.

It is straightforward to extract *initial rules*. We first generate all fully lexicalized source and target tree sequences using a dynamic programming algorithm and then iterate over all generated source and target tree sequence pairs  $\langle TS(f_{j_1}^{j_2}), TS(e_{i_1}^{i_2}) \rangle$ . If the condition " $\forall (i, j) \in A : i_1 \leq i \leq i_2 \leftrightarrow j_1 \leq j \leq j_2$ " is satisfied, the triple  $\langle TS(f_{j_1}^{j_2}), TS(e_{i_1}^{i_2}), \tilde{A} \rangle$  is an *initial rule*, where  $\tilde{A}$  are alignments between leaf nodes of  $TS(f_{j_1}^{j_2})$  and  $TS(e_{i_1}^{i_2})$ . We then derive *abstract rules* from *initial rules* by removing one or more of its *sub initial rules*. The *abstract rule* extraction algorithm presented next is implemented using dynamic programming. Due to space limitation, we skip the details here. In order to control the number of rules, we set three constraints for both finally extracted initial and abstract rules:

- 1) The depth of a tree in a rule is not greater

than  $h$ .

- 2) The number of non-terminals as leaf nodes is not greater than  $c$ .
- 3) The tree number in a rule is not greater than  $d$ .

In addition, we limit *initial rules* to have at most seven lexical words as leaf nodes on either side. However, in order to extract long-distance reordering rules, we also generate those *initial rules* with more than seven lexical words for *abstract rules* extraction only (not used in decoding). This makes our *abstract rules* more powerful in handling global structure reordering. Moreover, by configuring these parameters we can implement other translation models easily: 1) STSG-based model when  $d = 1$ ; 2) SCFG-based model when  $d = 1$  and  $h = 2$ ; 3) phrase-based translation model only (no reordering model) when  $c = 0$  and  $h = 1$ .

---

#### Algorithm 1: *abstract rules* extraction

---

**Input:** *initial rule* set  $r_{ini}$

**Output:** *abstract rule* set  $r_{abs}$

---

- 1: **for each**  $r_i \in r_{ini}$ , **do**
  - 2:   put all *sub initial rules* of  $r_i$  into a set  $r_i^{subini}$
  - 3:   **for each** subset  $\Theta \subseteq r_i^{subini}$  **do**
  - 4:     **if** there are spans overlapping between any two rules in the subset  $\Theta$  **then**
  - 5:       **continue** //go to line 3
  - 6:     **end if**
  - 7:     generate an abstract rule by removing the portions covered by  $\Theta$  from  $r_i$  and co-indexing the pairs of non-terminals that rooting the removed source and target parts
  - 8:     add them into the *abstract rule* set  $r_{abs}$
  - 9:   **end do**
  - 10: **end do**
- 

## 5 Decoding

Given  $T(f_1^J)$ , the decoder is to find the best derivation  $\theta$  that generates  $\langle T(f_1^J), T(e_1^I) \rangle$ .

$$\begin{aligned} \hat{e} &= \arg \max_{e_1^I} Pr(T(e_1^I) | T(f_1^J)) \\ &\approx \arg \max_{e_1^I, \theta} \prod_{r_i \in \theta} p(r_i) \end{aligned} \quad (4)$$

---

#### Algorithm 2: *Tree Sequence-based Decoder*

---

**Input:**  $T(f_1^J)$

**Output:**  $T(e_1^I)$

---

**Data structures:**

$h[J_1, J_2]$  To store translations to a span  $[J_1, J_2]$

- 1: **for**  $s = 0$  to  $J - 1$  **do** //  $s$ : span length
  - 2:   **for**  $j_1 = 1$  to  $J - s$ ,  $j_2 = j_1 + s$  **do**
  - 3:     **for each** rule  $r$  spanning  $[j_1, j_2]$  **do**
  - 4:       **if**  $r$  is an *initial rule* **then**
  - 5:         insert  $r$  into  $h[J_1, J_2]$
  - 6:       **else**
  - 7:         generate new translations from  $r$  by replacing non-terminal leaf nodes of  $r$  with their corresponding spans' translations that are already translated in previous steps
  - 8:         insert them into  $h[j_1, j_2]$
  - 9:       **end if**
  - 10:     **end for**
  - 11:   **end for**
  - 12: **end for**
  - 13: output the hypothesis with the highest score in  $h[1, J]$  as the final best translation
- 

The decoder is a span-based beam search together with a function for mapping the source derivations to the target ones. Algorithm 2 illustrates the decoding algorithm. It translates each span iteratively from small one to large one (lines 1-2). This strategy can guarantee that when translating the current span, all spans smaller than the current one have already been translated before if they are translatable (line 7). When translating a span, if the usable rule is an *initial rule*, then the tree sequence on the target side of the rule is a candidate translation (lines 4-5). Otherwise, we replace the non-terminal leaf nodes of the current *abstract rule* with their corresponding spans' translations that are already translated in previous steps (line 7). To speed up the decoder, we use several thresholds to limit search beams for each span:

- 1)  $\alpha$ , the maximal number of rules used
- 2)  $\beta$ , the minimal log probability of rules
- 3)  $\gamma$ , the maximal number of translations yield

It is worth noting that the decoder does not force a complete target parse tree to be generated. If no rules can be used to generate a complete target parse tree, the decoder just outputs whatever have

been translated so far monotonically as one hypothesis.

## 6 Experiments

### 6.1 Experimental Settings

We conducted Chinese-to-English translation experiments. We trained the translation model on the FBIS corpus (7.2M+9.2M words) and trained a 4-gram language model on the Xinhua portion of the English Gigaword corpus (181M words) using the SRILM Toolkits (Stolcke, 2002) with modified Kneser-Ney smoothing. We used sentences with less than 50 characters from the NIST MT-2002 test set as our development set and the NIST MT-2005 test set as our test set. We used the Stanford parser (Klein and Manning, 2003) to parse bilingual sentences on the training set and Chinese sentences on the development and test sets. The evaluation metric is case-sensitive BLEU-4 (Papineni et al., 2002). We used GIZA++ (Och and Ney, 2004) and the heuristics “grow-diag-final” to generate *m-to-n* word alignments. For the MER training (Och, 2003), we modified Koehn’s MER trainer (Koehn, 2004) for our tree sequence-based system. For significance test, we used Zhang et al’s implementation (Zhang et al, 2004).

We set three baseline systems: Moses (Koehn et al., 2007), and SCFG-based and STSG-based tree-to-tree translation models (Zhang et al., 2007). For Moses, we used its default settings. For the SCFG/STSG and our proposed model, we used the same settings except for the parameters  $d$  and  $h$  ( $d=1$  and  $h=2$  for the SCFG;  $d=1$  and  $h=6$  for the STSG;  $d=4$  and  $h=6$  for our model). We optimized these parameters on the training and development sets:  $c=3$ ,  $\alpha=20$ ,  $\beta=-100$  and  $\gamma=100$ .

### 6.2 Experimental Results

We carried out a number of experiments to examine the proposed tree sequence alignment-based translation model. In this subsection, we first report the rule distributions and compare our model with the three baseline systems. Then we study the model’s expressive ability by comparing the contributions made by different kinds of rules, including *strict* tree sequence rules, non-syntactic phrase rules, structure reordering rules and discontinuous

phrase rules<sup>2</sup>. Finally, we investigate the impact of maximal sub-tree number and sub-tree depth in our model. All of the following discussions are held on the training and test data.

Rule	Initial Rules		Abstract Rules	
	L	P	U	Total
<b>BP</b>	322,965	0	0	322,965
<b>TR</b>	443,010	144,459	24,871	612,340
<b>TSR</b>	225,570	103,932	714	330,216

Table 1: # of rules used in the testing ( $d=4$ ,  $h=6$ ) (**BP**: bilingual phrase (used in Moses), **TR**: tree rule (only 1 tree), **TSR**: tree sequence rule (> 1 tree), **L**: fully lexicalized, **P**: partially lexicalized, **U**: unlexicalized)

Table 1 reports the statistics of rules used in the experiments. It shows that:

1) We verify that the **BPs** are fully covered by the *initial rules* (i.e. lexicalized rules), in which the lexicalized **TSRs** model all non-syntactic phrase pairs with rich syntactic information. In addition, we find that the number of *initial rules* is greater than that of bilingual phrases. This is because one bilingual phrase can be covered by more than one *initial rule* which having different sub-tree structures.

2) *Abstract rules* generalize *initial rules* to unseen data and with structure reordering ability. The number of the *abstract rule* is far less than that of the *initial rules*. This is because leaf nodes of an *abstract rule* can be non-terminals that can represent any sub-trees using the non-terminals as roots.

Fig. 4 compares the performance of different models. It illustrates that:

1) Our tree sequence-based model significantly outperforms ( $p < 0.01$ ) previous phrase-based and linguistically syntax-based methods. This empirical-ly verifies the effect of the proposed method.

2) Both our method and STSG outperform Moses significantly. Our method also clearly outperforms STSG. These results suggest that:

- The linguistically motivated structure features are very useful for SMT, which can be cap-

<sup>2</sup> To be precise, we examine the contributions of *strict* tree sequence rules and single tree rules separately in this section. Therefore, unless specified, the term “tree sequence rules” used in this section only refers to the *strict* tree sequence rules, which must contain at least two sub-trees on the source side.



tured by the two syntax-based models through tree node operations.

- Our model is much more effective in utilizing linguistic structures than STSG since it uses tree sequence as basic translation unit. This allows our model not only to handle structure reordering by tree node operations in a larger span, but also to capture non-syntactic phrases, which circumvents previous syntactic constraints, thus giving our model more expressive power.

3) The linguistically motivated SCFG shows much lower performance. This is largely because SCFG only allows sibling nodes reordering and fails to utilize both non-syntactic phrases and those syntactic phrases that cannot be covered by a single CFG rule. It thereby suggests that SCFG is less effective in modelling parse tree structure transfer between Chinese and English when using Penn Treebank style linguistic grammar and under word-alignment constraints. However, formal SCFG show much better performance in the formally syntax-based translation framework (Chiang, 2005). This is because the formal syntax is learned from phrases directly without relying on any linguistic theory (Chiang, 2005). As a result, it is more robust to the issue of non-syntactic phrase usage and non-isomorphic structure alignment.

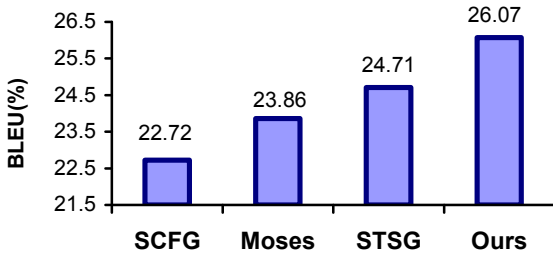


Figure 4: Performance comparison of different methods

Rule Type	TR (STSG)	TR +TSR_L	TR+TSR_L +TSR_P	TR +TSR
BLEU(%)	24.71	25.72	25.93	26.07

Table 2: Contributions of **TSRs** (see Table 1 for the definitions of the abbreviations used in this table)

Table 2 measures the contributions of different kinds of tree sequence rules. It suggests that:

1) All the three kinds of **TSRs** contribute to the performance improvement and their combination

further improves the performance. It suggests that they are complementary to each other since the lexicalized **TSRs** are used to model non-syntactic phrases while the other two kinds of **TSRs** can generalize the lexicalized rules to unseen phrases.

2) The lexicalized **TSRs** make the major contribution since they can capture non-syntactic phrases with syntactic structure features.

Rule Type	BLEU (%)
<b>TR+TSR</b>	26.07
( <b>TR+TSR</b> ) w/o <b>SRR</b>	24.62
( <b>TR+TSR</b> ) w/o <b>DPR</b>	25.78

Table 3: Effect of **Structure Reordering Rules (SRR)**: refers to the structure reordering rules that have at least two non-terminal leaf nodes with inverted order in the source and target sides, which are usually not captured by phrase-based models. Note that the reordering between lexical words and non-terminal leaf nodes is not considered here) and **Discontinuous Phrase Rules (DPR)**: refers to these rules having at least one non-terminal leaf node between two lexicalized leaf nodes) in our tree sequence-based model ( $d = 4$  and  $h = 6$ )

Rule Type	# of rules	# of rules overlapped (Intersection)
SRR	68,217	18,379 (26.9%)
DPR	57,244	18,379 (32.1%)

Table 4: numbers of SRR and DPR rules

Table 3 shows the contributions of SRR and DPR. It clearly indicates that SRRs are very effective in reordering structures, which improve performance by 1.45 (26.07-24.62) BLEU score. However, DPRs have less impact on performance in our tree sequence-based model. This seems in contradiction to the previous observations<sup>3</sup> in literature. However, it is not surprising simply because we use tree sequences as the basic translation units. Thereby, our model can capture all phrases. In this sense, our model behaves like a phrase-based model, less sensitive to discontinuous phras-

<sup>3</sup> Wellington et al. (2006) reports that discontinuities are very useful for translational equivalence analysis using binary-branching structures under word alignment and parse tree constraints while they are almost of no use if under word alignment constraints only. Bod (2007) finds that discontinuous phrase rules make significant performance improvement in linguistically STSG-based SMT models.

es (Wellington et al., 2006). Our additional experiments also verify that discontinuous phrase rules are complementary to syntactic phrase rules (Bod, 2007) while non-syntactic phrase rules may compromise the contribution of discontinuous phrase rules. Table 4 reports the numbers of these two kinds of rules. It shows that around 30% rules are shared by the two kinds of rule sets. These overlapped rules contain at least two non-terminal leaf nodes plus two terminal leaf nodes, which implies that longer rules do not affect performance too much.

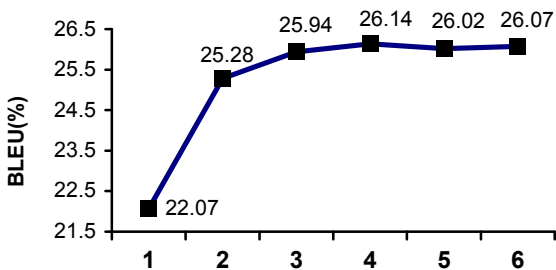


Figure 5: Accuracy changing with different maximal tree depths ( $h = 1$  to  $6$  when  $d = 4$ )

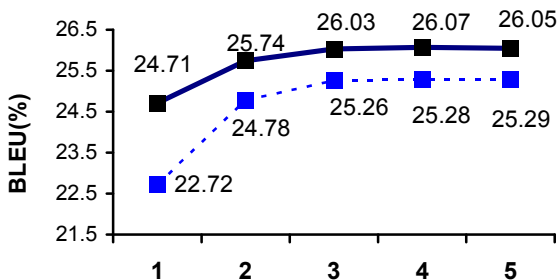


Figure 6: Accuracy changing with the different maximal number of trees in a tree sequence ( $d = 1$  to  $5$ ), the upper line is for  $h = 6$  while the lower line is for  $h = 2$ .

Fig. 5 studies the impact when setting different maximal tree depth ( $h$ ) in a rule on the performance. It demonstrates that:

1) Significant performance improvement is achieved when the value of  $h$  is increased from 1 to 2. This can be easily explained by the fact that when  $h = 1$ , only monotonic search is conducted, while  $h = 2$  allows non-terminals to be leaf nodes, thus introducing preliminary structure features to the search and allowing non-monotonic search.

2) Internal structures and large span (due to  $h$  increasing) are also useful as attested by the gain

of 0.86 (26.14-25.28) Blue score when the value of  $h$  increases from 2 to 4.

Fig. 6 studies the impact on performance by setting different maximal tree number ( $d$ ) in a rule. It further indicates that:

1) Tree sequence rules ( $d > 1$ ) are useful and even more helpful if we limit the tree depth to no more than two (lower line,  $h=2$ ). However, tree sequence rules consisting of more than three sub-trees have almost no contribution to the performance improvement. This is mainly due to data sparseness issue when  $d > 3$ .

2) Even if only two-layer sub-trees (lower line) are allowed, our method still outperforms STSG and Moses when  $d > 1$ . This further validates the effectiveness of our design philosophy of using multi-sub-trees as basic translation unit in SMT.

## 7 Conclusions and Future Work

In this paper, we present a tree sequence alignment-based translation model to combine the strengths of phrase-based and syntax-based methods. The experimental results on the NIST MT-2005 Chinese-English translation task demonstrate the effectiveness of the proposed model. Our study also finds that in our model the tree sequence rules are very useful since they can model non-syntactic phrases and reorderings with rich linguistic structure features while discontinuous phrases and tree sequence rules with more than three sub-trees have less impact on performance.

There are many interesting research topics on the tree sequence-based translation model worth exploring in the future. The current method extracts large amount of rules. Many of them are redundant, which make decoding very slow. Thus, effective rule optimization and pruning algorithms are highly desirable. Ideally, a linguistically and empirically motivated theory can be worked out, suggesting what kinds of rules should be extracted given an input phrase pair. For example, most function words and headwords can be kept in *abstract rules* as features. In addition, word alignment is a hard constraint in our rule extraction. We will study direct structure alignments to reduce the impact of word alignment errors. We are also interested in comparing our method with the forest-to-string model (Liu et al., 2007). Finally, we would also like to study unsupervised learning-based bilingual parsing for SMT.

## References

- Rens Bod. 2007. *Unsupervised Syntax-Based Machine Translation: The Contribution of Discontinuous Phrases*. MT-Summit-07. 51-56.
- David Chiang. 2005. *A hierarchical phrase-based model for SMT*. ACL-05. 263-270.
- Brooke Cowan, Ivona Kucerova and Michael Collins. 2006. *A discriminative model for tree-to-tree translation*. EMNLP-06. 232-241.
- Yuan Ding and Martha Palmer. 2005. *Machine translation using probabilistic synchronous dependency insertion grammars*. ACL-05. 541-548.
- Jason Eisner. 2003. *Learning non-isomorphic tree mappings for MT*. ACL-03 (companion volume).
- Michel Galley, Mark Hopkins, Kevin Knight and Daniel Marcu. 2004. *What's in a translation rule?* HLT-NAACL-04.
- Michel Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang and I. Thayer. 2006. *Scalable Inference and Training of Context-Rich Syntactic Translation Models*. COLING-ACL-06. 961-968
- Daniel Gildea. 2003. *Loosely Tree-Based Alignment for Machine Translation*. ACL-03. 80-87.
- Jonathan Graehl and Kevin Knight. 2004. *Training tree transducers*. HLT-NAACL-2004. 105-112.
- Mary Hearne and Andy Way. 2003. *Seeing the wood for the trees: data-oriented translation*. MT Summit IX, 165-172.
- Liang Huang, Kevin Knight and Aravind Joshi. 2006. *Statistical Syntax-Directed Translation with Extended Domain of Locality*. AMTA-06 (poster).
- Dan Klein and Christopher D. Manning. 2003. *Accurate Unlexicalized Parsing*. ACL-03. 423-430.
- Philipp Koehn, F. J. Och and D. Marcu. 2003. *Statistical phrase-based translation*. HLT-NAACL-03. 127-133.
- Philipp Koehn. 2004. *Pharaoh: a beam search decoder for phrase-based statistical machine translation models*. AMTA-04, 115-124
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst. 2007. *Moses: Open Source Toolkit for Statistical Machine Translation*. ACL-07 (poster) 77-180.
- Yang Liu, Qun Liu and Shouxun Lin. 2006. *Tree-to-String Alignment Template for Statistical Machine Translation*. COLING-ACL-06. 609-616.
- Yang Liu, Yun Huang, Qun Liu and Shouxun Lin. 2007. *Forest-to-String Statistical Translation Rules*. ACL-07. 704-711.
- Daniel Marcu, W. Wang, A. Echihabi and K. Knight. 2006. *SPMT: Statistical Machine Translation with Syntactified Target Language Phrases*. EMNLP-06. 44-52.
- I. Dan Melamed. 2004. *Statistical machine translation by parsing*. ACL-04. 653-660.
- Franz J. Och and Hermann Ney. 2002. *Discriminative training and maximum entropy models for statistical machine translation*. ACL-02. 295-302.
- Franz J. Och. 2003. *Minimum error rate training in statistical machine translation*. ACL-03. 160-167.
- Franz J. Och and Hermann Ney. 2004a. *The alignment template approach to statistical machine translation*. Computational Linguistics, 30(4):417-449.
- Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. *BLEU: a method for automatic evaluation of machine translation*. ACL-02. 311-318.
- Arjen Poutsma. 2000. *Data-oriented translation*. COLING-2000. 635-641
- Chris Quirk and Arul Menezes. 2006. *Do we need phrases? Challenging the conventional wisdom in SMT*. COLING-ACL-06. 9-16.
- Chris Quirk, Arul Menezes and Colin Cherry. 2005. *Dependency treelet translation: Syntactically informed phrasal SMT*. ACL-05. 271-279.
- Stefan Riezler and John T. Maxwell III. 2006. *Grammatical Machine Translation*. HLT-NAACL-06. 248-255.
- Hendra Setiawan, Min-Yen Kan and Haizhou Li. 2007. *Ordering Phrases with Function Words*. ACL-7. 712-719.
- Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit*. ICSLP-02. 901-904.
- Benjamin Wellington, Sonjia Waxmonsky and I. Dan Melamed. 2006. *Empirical Lower Bounds on the Complexity of Translational Equivalence*. COLING-ACL-06. 977-984.
- Dekai Wu. 1997. *Stochastic inversion transduction grammars and bilingual parsing of parallel corpora*. Computational Linguistics, 23(3):377-403.
- Deyi Xiong, Qun Liu and Shouxun Lin. 2006. *Maximum Entropy Based Phrase Reordering Model for SMT*. COLING-ACL-06. 521- 528.
- Kenji Yamada and Kevin Knight. 2001. *A syntax-based statistical translation model*. ACL-01. 523-530.
- Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Sheng Li and Chew Lim Tan. 2007. *A Tree-to-Tree Alignment-based Model for Statistical Machine Translation*. MT-Summit-07. 535-542.
- Ying Zhang, Stephan Vogel, Alex Waibel. 2004. *Interpreting BLEU/NIST scores: How much improvement do we need to have a better system?* LREC-04. 2051-2054.

# Automatic Syllabification with Structured SVMs for Letter-To-Phoneme Conversion

Susan Bartlett<sup>†</sup>

Grzegorz Kondrak<sup>†</sup>

Colin Cherry<sup>‡</sup>

<sup>†</sup>Department of Computing Science  
University of Alberta  
Edmonton, AB, T6G 2E8, Canada

<sup>‡</sup>Microsoft Research  
One Microsoft Way  
Redmond, WA, 98052

{susan,kondrak}@cs.ualberta.ca colinc@microsoft.com

## Abstract

We present the first English syllabification system to improve the accuracy of letter-to-phoneme conversion. We propose a novel discriminative approach to automatic syllabification based on structured SVMs. In comparison with a state-of-the-art syllabification system, we reduce the syllabification word error rate for English by 33%. Our approach also performs well on other languages, comparing favorably with published results on German and Dutch.

## 1 Introduction

Pronouncing an unfamiliar word is a task that is often accomplished by breaking the word down into smaller components. Even small children learning to read are taught to pronounce a word by “sounding out” its parts. Thus, it is not surprising that Letter-to-Phoneme (L2P) systems, which convert orthographic forms of words into sequences of phonemes, can benefit from subdividing the input word into smaller parts, such as syllables or morphemes. Marchand and Damer (2007) report that incorporating oracle syllable boundary information improves the accuracy of their L2P system, but they fail to emulate that result with any of their automatic syllabification methods. Demberg et al. (2007), on the other hand, find that morphological segmentation boosts L2P performance in German, but not in English. To our knowledge, no previous English orthographic syllabification system has been able to actually improve performance on the larger L2P problem.

In this paper, we focus on the task of automatic orthographic syllabification, with the explicit goal

of improving L2P accuracy. A syllable is a subdivision of a word, typically consisting of a vowel, called the nucleus, and the consonants preceding and following the vowel, called the onset and the coda, respectively. Although in the strict linguistic sense syllables are phonological rather than orthographic entities, our L2P objective constrains the input to orthographic forms. Syllabification of phonemic representation is in fact an easier task, which we plan to address in a separate publication.

Orthographic syllabification is sometimes referred to as *hyphenation*. Many dictionaries provide hyphenation information for orthographic word forms. These hyphenation schemes are related to, and influenced by, phonemic syllabification. They serve two purposes: to indicate where words may be broken for end-of-line divisions, and to assist the dictionary reader with correct pronunciation (Gove, 1993). Although these purposes are not always consistent with our objective, we show that we can improve L2P conversion by taking advantage of the available hyphenation data. In addition, automatic hyphenation is a legitimate task by itself, which could be utilized in word editors or in synthesizing new trade names from several concepts.

We present a discriminative approach to orthographic syllabification. We formulate syllabification as a tagging problem, and learn a discriminative tagger from labeled data using a structured support vector machine (SVM) (Tsochantaridis et al., 2004). With this approach, we reduce the error rate for English by 33%, relative to the best existing system. Moreover, we are also able to improve a state-of-the-art L2P system by incorporating our syllabification models. Our method is not language specific; when applied to German and Dutch, our performance is

comparable with the best existing systems in those languages, even though our system has been developed and tuned on English only.

The paper is structured as follows. After discussing previous computational approaches to the problem (Section 2), we introduce structured SVMs (Section 3), and outline how we apply them to orthographic syllabification (Section 4). We present our experiments and results for the syllabification task in Section 5. In Section 6, we apply our syllabification models to the L2P task. Section 7 concludes.

## 2 Related Work

Automatic preprocessing of words is desirable because the productive nature of language ensures that no finite lexicon will contain all words. Marchand et al. (2007) show that rule-based methods are relatively ineffective for orthographic syllabification in English. On the other hand, few data-driven syllabification systems currently exist.

Demberg (2006) uses a fourth-order Hidden Markov Model to tackle orthographic syllabification in German. When added to her L2P system, Demberg’s orthographic syllabification model effects a one percent absolute improvement in L2P word accuracy.

Bouma (2002) explores syllabification in Dutch. He begins with finite state transducers, which essentially implement a general preference for onsets. Subsequently, he uses transformation-based learning to automatically extract rules that improve his system. Bouma’s best system, trained on some 250K examples, achieves 98.17% word accuracy. Daelemans and van den Bosch (1992) implement a back-propagation network for Dutch orthography, but find it is outperformed by less complex look-up table approaches.

Marchand and Damper (2007) investigate the impact of syllabification on the L2P problem in English. Their Syllabification by Analogy (SbA) algorithm is a data-driven, lazy learning approach. For each input word, SbA finds the most similar substrings in a lexicon of syllabified words and then applies these dictionary syllabifications to the input word. Marchand and Damper report 78.1% word accuracy on the NETtalk dataset, which is not good enough to improve their L2P system.

Chen (2003) uses an n-gram model and Viterbi decoder as a syllabifier, and then applies it as a pre-processing step in his maximum-entropy-based English L2P system. He finds that the syllabification pre-processing produces no gains over his baseline system.

Marchand et al. (2007) conduct a more systematic study of existing syllabification approaches. They examine syllabification in both the pronunciation and orthographic domains, comparing their own SbA algorithm with several instance-based learning approaches (Daelemans et al., 1997; van den Bosch, 1997) and rule-based implementations. They find that SbA universally outperforms these other approaches by quite a wide margin.

Syllabification of phonemes, rather than letters, has also been investigated (Müller, 2001; Pearson et al., 2000; Schmid et al., 2007). In this paper, our focus is on orthographic forms. However, as with our approach, some previous work in the phonetic domain has formulated syllabification as a tagging problem.

## 3 Structured SVMs

A structured support vector machine (SVM) is a large-margin training method that can learn to predict structured outputs, such as tag sequences or parse trees, instead of performing binary classification (Tsochantaridis et al., 2004). We employ a structured SVM that predicts tag sequences, called an SVM Hidden Markov Model, or SVM-HMM. This approach can be considered an HMM because the Viterbi algorithm is used to find the highest scoring tag sequence for a given observation sequence. The scoring model employs a Markov assumption: each tag’s score is modified only by the tag that came before it. This approach can be considered an SVM because the model parameters are trained discriminatively to separate correct tag sequences from incorrect ones by as large a margin as possible. In contrast to generative HMMs, the learning process requires labeled training data.

There are a number of good reasons to apply the structured SVM formalism to this problem. We get the benefit of discriminative training, not available in a generative HMM. Furthermore, we can use an arbitrary feature representation that does not require

any conditional independence assumptions. Unlike a traditional SVM, the structured SVM considers complete tag sequences during training, instead of breaking each sequence into a number of training instances.

Training a structured SVM can be viewed as a multi-class classification problem. Each training instance  $x_i$  is labeled with a correct tag sequence  $y_i$  drawn from a set of possible tag sequences  $Y_i$ . As is typical of discriminative approaches, we create a feature vector  $\Psi(x, y)$  to represent a candidate  $y$  and its relationship to the input  $x$ . The learner’s task is to weight the features using a vector  $w$  so that the correct tag sequence receives more weight than the competing, incorrect sequences:

$$\forall_i \forall_{y \in Y_i, y \neq y_i} [\Psi(x_i, y_i) \cdot w > \Psi(x_i, y) \cdot w] \quad (1)$$

Given a trained weight vector  $w$ , the SVM tags new instances  $x_i$  according to:

$$\operatorname{argmax}_{y \in Y_i} [\Psi(x_i, y) \cdot w] \quad (2)$$

A structured SVM finds a  $w$  that satisfies Equation 1, and separates the correct taggings by as large a margin as possible. The  $\operatorname{argmax}$  in Equation 2 is conducted using the Viterbi algorithm.

Equation 1 is a simplification. In practice, a structured distance term is added to the inequality in Equation 1 so that the required margin is larger for tag sequences that diverge further from the correct sequence. Also, slack variables are employed to allow a trade-off between training accuracy and the complexity of  $w$ , via a tunable cost parameter.

For most structured problems, the set of negative sequences in  $Y_i$  is exponential in the length of  $x_i$ , and the constraints in Equation 1 cannot be explicitly enumerated. The structured SVM solves this problem with an iterative online approach:

1. Collect the most damaging incorrect sequence  $y$  according to the current  $w$ .
2. Add  $y$  to a growing set  $\bar{Y}_i$  of incorrect sequences.
3. Find a  $w$  that satisfies Equation 1, using the partial  $\bar{Y}_i$  sets in place of  $Y_i$ .
4. Go to next training example, loop to step 1.

This iterative process is explained in far more detail in (Tsochantaridis et al., 2004).

## 4 Syllabification with Structured SVMs

In this paper we apply structured SVMs to the syllabification problem. Specifically, we formulate syllabification as a tagging problem and apply the SVM-HMM software package<sup>1</sup> (Altun et al., 2003). We use a linear kernel, and tune the SVM’s cost parameter on a development set. The feature representation  $\Psi$  consists of emission features, which pair an aspect of  $x$  with a single tag from  $y$ , and transition features, which count tag pairs occurring in  $y$ . With SVM-HMM, the crux of the task is to create a tag scheme and feature set that produce good results. In this section, we discuss several different approaches to tagging for the syllabification task. Subsequently, we outline our emission feature representation. While developing our tagging schemes and feature representation, we used a development set of 5K words held out from our CELEX training data. All results reported in this section are on that set.

### 4.1 Annotation Methods

We have employed two different approaches to tagging in this research. **Positional tags** capture where a letter occurs within a syllable; **Structural tags** express the role each letter is playing within the syllable.

#### Positional Tags

The **NB tag** scheme simply labels every letter as either being at a syllable boundary (B), or not (N). Thus, the word *im-mor-al-ly* is tagged  $\langle N B N N B N B N N \rangle$ , indicating a syllable boundary after each *B* tag. This binary classification approach to tagging is implicit in several previous implementations (Daelemans and van den Bosch, 1992; Bouma, 2002), and has been done explicitly in both the orthographic (Demberg, 2006) and phoneme domains (van den Bosch, 1997).

A weakness of NB tags is that they encode no knowledge about the length of a syllable. Intuitively, we expect the length of a syllable to be valuable information — most syllables in English contain fewer than four characters. We introduce a tagging scheme that sequentially numbers the *N* tags to impart information about syllable length. Under the **Numbered**

<sup>1</sup>[http://svmlight.joachims.org/svm\\_struct.html](http://svmlight.joachims.org/svm_struct.html)

**NB tag** scheme, *im-mor-al-ly* is annotated as  $\langle N1 B N1 N2 B N1 B N1 N2 \rangle$ . With this tag set, we have effectively introduced a bias in favor of shorter syllables: tags like *N6*, *N7*... are comparatively rare, so the learner will postulate them only when the evidence is particularly compelling.

### Structural Tags

Numbered NB tags are more informative than standard NB tags. However, neither annotation system can represent the internal structure of the syllable. This has advantages: tags can be automatically generated from a list of syllabified words without even a passing familiarity with the language. However, a more informative annotation, tied to phonotactics, ought to improve accuracy. Krenn (1997) proposes the **ONC tag** scheme, in which phonemes of a syllable are tagged as an onset, nucleus, or coda. Given these ONC tags, syllable boundaries can easily be generated by applying simple regular expressions.

Unfortunately, it is not as straightforward to generate ONC-tagged training data in the orthographic domain, even with syllabified training data. Silent letters are problematic, and some letters can behave differently depending on their context (in English, consonants such as *m*, *y*, and *l* can act as vowels in certain situations). Thus, it is difficult to generate ONC tags for orthographic forms without at least a cursory knowledge of the language and its principles.

For English, tagging the syllabified training set with ONC tags is performed by the following simple algorithm. In the first stage, all letters from the set  $\{a, e, i, o, u\}$  are marked as vowels, while the remaining letters are marked as consonants. Next, we examine all the instances of the letter *y*. If a *y* is both preceded and followed by a consonant, we mark that instance as a vowel rather than a consonant. In the second stage, the first group of consecutive vowels in each syllable is tagged as nucleus. All letters preceding the nucleus are then tagged as onset, while all letters following the nucleus are tagged as coda.

Our development set experiments suggested that numbering ONC tags increases their performance. Under the **Numbered ONC tag** scheme, the single-syllable word *stealth* is labeled  $\langle O1 O2 N1 N2 C1 C2 C3 \rangle$ .

A disadvantage of Numbered ONC tags is that, unlike positional tags, they do not represent syllable breaks explicitly. Within the ONC framework, we need the conjunction of two tags (such as an *N1* tag followed by an *O1* tag) to represent the division between syllables. This drawback can be overcome by combining ONC tags and NB tags in a hybrid **Break ONC tag** scheme. Using Break ONC tags, the word *lev-i-ty* is annotated as  $\langle O N C B N B O N \rangle$ . The  $\langle NB \rangle$  tag indicates a letter is both part of the nucleus and before a syllable break, while the  $\langle N \rangle$  tag represents a letter that is part of a nucleus but in the middle of a syllable. In this way, we get the best of both worlds: tags that encapsulate information about syllable structure, while also representing syllable breaks explicitly with a single tag.

### 4.2 Emission Features

SVM-HMM predicts a tag for each letter in a word, so emission features use aspects of the input to help predict the correct tag for a specific letter. Consider the tag for the letter *o* in the word *immorally*. With a traditional HMM, we consider only that it is an *o* being emitted, and assess potential tags based on that single letter. The SVM framework is less restrictive: we can include *o* as an emission feature, but we can also include features indicating that the preceding and following letters are *m* and *r* respectively. In fact, there is no reason to confine ourselves to only one character on either side of the focus letter.

After experimenting with the development set, we decided to include in our feature set a window of eleven characters around the focus character, five on either side. Figure 1 shows that performance gains level off at this point. Special beginning- and end-of-word characters are appended to words so that every letter has five characters before and after. We also experimented with asymmetric context windows, representing more characters after the focus letter than before, but we found that symmetric context windows perform better.

Because our learner is effectively a linear classifier, we need to explicitly represent any important conjunctions of features. For example, the bigram *bl* frequently occurs within a single English syllable, while the bigram *lb* generally straddles two syllables. Similarly, a fourgram like *tion* very often

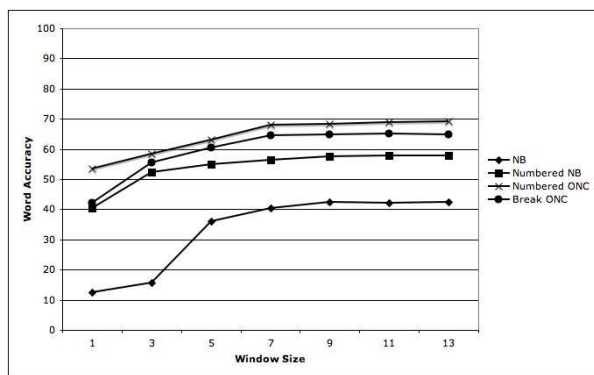


Figure 1: Word accuracy as a function of the window size around the focus character, using unigram features on the development set.

forms a syllable in and of itself. Thus, in addition to the single-letter features outlined above, we also include in our representation any bigrams, trigrams, four-grams, and five-grams that fit inside our context window. As is apparent from Figure 2, we see a substantial improvement by adding bigrams to our feature set. Higher-order n-grams produce increasingly smaller gains.

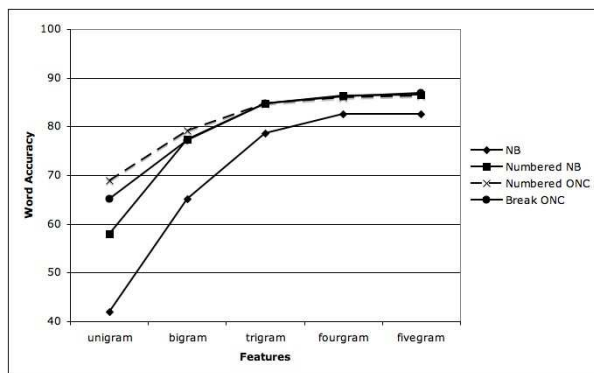


Figure 2: Word accuracy as a function of maximum n-gram size on the development set.

In addition to these primary n-gram features, we experimented with linguistically-derived features. Intuitively, basic linguistic knowledge, such as whether a letter is a consonant or a vowel, should be helpful in determining syllabification. However, our experiments suggested that including features like these has no significant effect on performance. We believe that this is caused by the ability of the SVM to learn such generalizations from the n-gram features alone.

## 5 Syllabification Experiments

In this section, we will discuss the results of our best emission feature set (five-gram features with a context window of eleven letters) on held-out unseen test sets. We explore several different languages and datasets, and perform a brief error analysis.

### 5.1 Datasets

Datasets are especially important in syllabification tasks. Dictionaries sometimes disagree on the syllabification of certain words, which makes a gold standard difficult to obtain. Thus, any reported accuracy is only with respect to a given set of data.

In this paper, we report the results of experiments on two datasets: CELEX and NETtalk. We focus mainly on CELEX, which has been developed over a period of years by linguists in the Netherlands. CELEX contains English, German, and Dutch words, and their orthographic syllabifications. We removed all duplicates and multiple-word entries for our experiments. The NETtalk dictionary was originally developed with the L2P task in mind. The syllabification data in NETtalk was created manually in the phoneme domain, and then mapped directly to the letter domain.

NETtalk and CELEX do not provide the same syllabification for every word. There are numerous instances where the two datasets differ in a perfectly reasonable manner (*e.g. for-ging* in NETtalk vs. *forg-ing* in CELEX). However, we argue that NETtalk is a vastly inferior dataset. On a sample of 50 words, NETtalk agrees with Merriam-Webster's syllabifications in only 54% of instances, while CELEX agrees in 94% of cases. Moreover, NETtalk is riddled with truly bizarre syllabifications, such as *be-aver*, *dis-hcloth* and *som-ething*. These syllabifications make generalization very hard, and are likely to complicate the L2P task we ultimately want to accomplish. Because previous work in English primarily used NETtalk, we report our results on both datasets. Nevertheless, we believe NETtalk is unsuitable for building a syllabification model, and that results on CELEX are much more indicative of the efficacy of our (or any other) approach.

At 20K words, NETtalk is much smaller than CELEX. For NETtalk, we randomly divide the data into 13K training examples and 7K test words. We



randomly select a comparably-sized training set for our CELEX experiments (14K), but test on a much larger, 25K set. Recall that 5K training examples were held out as a development set.

## 5.2 Results

We report the results using two metrics. Word accuracy (WA) measures how many words match the gold standard. Syllable break error rate (SBER) captures the incorrect tags that cause an error in syllabification. Word accuracy is the more demanding metric. We compare our system to Syllabification by Analogy (SbA), the best existing system for English (Marchand and Damper, 2007). For both CELEX and NETtalk, SbA was trained and tested with the same data as our structured SVM approach.

Data Set	Method	WA	SBER
CELEX	NB tags	86.66	2.69
	Numbered NB	89.45	2.51
	Numbered ONC	89.86	2.50
	Break ONC	89.99	2.42
	SbA	84.97	3.96
NETtalk	Numbered NB	81.75	5.01
	SbA	75.56	7.73

Table 1: Syllabification performance in terms of word accuracy and syllable break error percentage.

Table 1 presents the word accuracy and syllable break error rate achieved by each of our tag sets on both the CELEX and NETtalk datasets. Of our four tag sets, NB tags perform noticeably worse. This is an important result because it demonstrates that it is not sufficient to simply model a syllable’s boundaries; we must also model a syllable’s length or structure to achieve the best results. Given the similarity in word accuracy scores, it is difficult to draw definitive conclusions about the remaining three tags sets, but it does appear that there is an advantage to modeling syllable structure, as both ONC tag sets score better than the best NB set.

All variations of our system outperform SbA on both datasets. Overall, our best tag set lowers the error rate by one-third, relative to SbA’s performance. Note that we employ only numbered NB tags for the NETtalk test; we could not apply structural tag schemes to the NETtalk training data because of its

bizarre syllabification choices.

Our higher level of accuracy is also achieved more efficiently. Once a model is learned, our system can syllabify 25K words in about a minute, while SbA requires several hours (Marchand, 2007). SVM training times vary depending on the tag set and dataset used, and the number of training examples. On 14K CELEX examples with the ONC tag set, our model trained in about an hour, on a single-processor P4 3.4GHz processor. Training time is, of course, a one-time cost. This makes our approach much more attractive for inclusion in an actual L2P system.

Figure 3 shows our method’s learning curve. Even small amounts of data produce adequate performance — with only 2K training examples, word accuracy is already over 75%. Using a 60K training set and testing on a held-out 5K set, we see word accuracies climb to 95.65%.

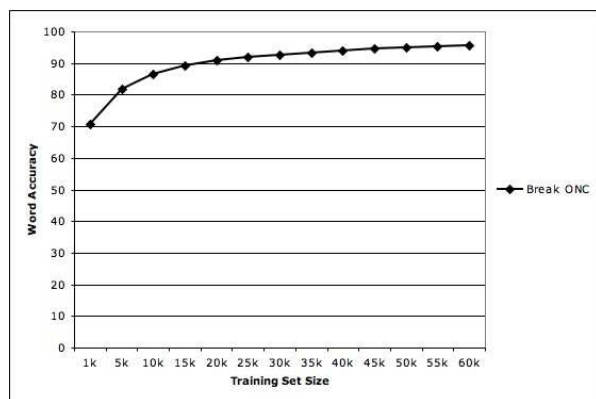


Figure 3: Word accuracy as function of the size of the training data.

## 5.3 Error Analysis

We believe that the reason for the relatively low performance of unnumbered NB tags is the weakness of the signal coming from NB emission features. With the exception of *q* and *x*, every letter can take on either an N tag or a B tag with almost equal probability. This is not the case with Numbered NB tags. Vowels are much more likely to have N2 or N3 tags (because they so often appear in the middle of a syllable), while consonants take on N1 labels with greater probability.

The numbered NB and ONC systems make many of the same errors, on words that we might expect to

cause difficulty. In particular, both suffer from being unaware of compound nouns and morphological phenomena. All three systems, for example, incorrectly syllabify *hold-o-ver* as *hol-dov-er*. This kind of error is caused by a lack of knowledge of the component words. The three systems also display trouble handling consecutive vowels, as when *co-ad-ju-tors* is syllabified incorrectly as *coad-ju-tors*. Vowel pairs such as *oa* are not handled consistently in English, and the SVM has trouble predicting the exceptions.

## 5.4 Other Languages

We take advantage of the language-independence of Numbered NB tags to apply our method to other languages. Without even a cursory knowledge of German or Dutch, we have applied our approach to these two languages.

# Data Points	Dutch	German
~ 50K	98.20	98.81
~ 250K	99.45	99.78

Table 2: Syllabification performance in terms of word accuracy percentage.

We have randomly selected two training sets from the German and Dutch portions of CELEX. Our smaller model is trained on ~ 50K words, while our larger model is trained on ~ 250K. Table 2 shows our performance on a 30K test set held out from both training sets. Results from both the small and large models are very good indeed.

Our performance on these language sets is clearly better than our best score for English (compare at 95% with a comparable amount of training data). Syllabification is a more regular process in German and Dutch than it is in English, which allows our system to score higher on those languages.

Our method’s word accuracy compares favorably with other methods. Bouma’s finite state approach for Dutch achieves 96.49% word accuracy using 50K training points, while we achieve 98.20%. With a larger model, trained on about 250K words, Bouma achieves 98.17% word accuracy, against our 99.45%. Demberg (2006) reports that her HMM approach for German scores 97.87% word accuracy, using a 90/10 training/test split on the CELEX

dataset. On the same set, Demberg et al. (2007) obtain 99.28% word accuracy by applying the system of Schmid et al. (2007). Our score using a similar split is 99.78%.

Note that none of these scores are directly comparable, because we did not use the same train-test splits as our competitors, just similar amounts of training and test data. Furthermore, when assembling random train-test splits, it is quite possible that words sharing the same lemma will appear in both the training and test sets. This makes the problem much easier with large training sets, where the chance of this sort of overlap becomes high. Therefore, any large data results may be slightly inflated as a prediction of actual out-of-dictionary performance.

## 6 L2P Performance

As we stated from the outset, one of our primary motivations for exploring orthographic syllabification is the improvements it can produce in L2P systems. To explore this, we tested our model in conjunction with a recent L2P system that has been shown to predict phonemes with state-of-the-art word accuracy (Jiampojarn et al., 2007). Using a model derived from training data, this L2P system first divides a word into letter chunks, each containing one or two letters. A local classifier then predicts a number of likely phonemes for each chunk, with confidence values. A phoneme-sequence Markov model is then used to select the most likely sequence from the phonemes proposed by the local classifier.

Syllabification	English	Dutch	German
None	84.67	91.56	90.18
Numbered NB	85.55	92.60	90.59
Break ONC	85.59	N/A	N/A
Dictionary	86.29	93.03	90.57

Table 3: Word accuracy percentage on the letter-to-phoneme task with and without the syllabification information.

To measure the improvement syllabification can effect on the L2P task, the L2P system was trained with syllabified, rather than unsyllabified words. Otherwise, the execution of the L2P system remains unchanged. Data for this experiment is again drawn

from the CELEX dictionary. In Table 3, we report the average word accuracy achieved by the L2P system using 10-fold cross-validation. We report L2P performance without any syllabification information, with perfect dictionary syllabification, and with our small learned models of syllabification. L2P performance with dictionary syllabification represents an approximate upper bound on the contributions of our system.

Our syllabification model improves L2P performance. In English, perfect syllabification produces a relative error reduction of 10.6%, and our model captures over half of the possible improvement, reducing the error rate by 6.0%. To our knowledge, this is the first time a syllabification model has improved L2P performance in English. Previous work includes Marchand and Damper (2007)’s experiments with SbA and the L2P problem on NETalk. Although perfect syllabification reduces their L2P relative error rate by 18%, they find that their learned model actually increases the error rate. Chen (2003) achieved word accuracy of 91.7% for his L2P system, testing on a different dictionary (Pronlex) with a much larger training set. He does not report word accuracy for his syllabification model. However, his baseline L2P system is not improved by adding a syllabification model.

For Dutch, perfect syllabification reduces the relative L2P error rate by 17.5%; we realize over 70% of the available improvement with our syllabification model, reducing the relative error rate by 12.4%.

In German, perfect syllabification produces only a small reduction of 3.9% in the relative error rate. Experiments show that our learned model actually produces a slightly higher reduction in the relative error rate. This anomaly may be due to errors or inconsistencies in the dictionary syllabifications that are not replicated in the model output. Previously, Demberg (2006) generated statistically significant L2P improvements in German by adding syllabification pre-processing. However, our improvements are coming at a much higher baseline level of word accuracy – 90% versus only 75%.

Our results also provide some evidence that syllabification preprocessing may be more beneficial to L2P than morphological preprocessing. Demberg et al. (2007) report that oracle morphological annotation produces a relative error rate reduction

of 3.6%. We achieve a larger decrease at a higher level of accuracy, using an automatic pre-processing technique. This may be because orthographic syllabifications already capture important facts about a word’s morphology.

## 7 Conclusion

We have applied structured SVMs to the syllabification problem, clearly outperforming existing systems. In English, we have demonstrated a 33% relative reduction in error rate with respect to the state of the art. We used this improved syllabification to increase the letter-to-phoneme accuracy of an existing L2P system, producing a system with 85.5% word accuracy, and recovering more than half of the potential improvement available from perfect syllabification. This is the first time automatic syllabification has been shown to improve English L2P. Furthermore, we have demonstrated the language-independence of our system by producing competitive orthographic syllabification solutions for both Dutch and German, achieving word syllabification accuracies of 98% and 99% respectively. These learned syllabification models also improve accuracy for German and Dutch letter-to-phoneme conversion.

In future work on this task, we plan to explore adding morphological features to the SVM, in an effort to overcome errors in compound words and inflectional forms. We would like to experiment with performing L2P and syllabification jointly, rather than using syllabification as a pre-processing step for L2P. We are also working on applying our method to phonetic syllabification.

## Acknowledgements

Many thanks to Sittichai Jiampoamarn for his help with the L2P experiments, and to Yannick Marchand for providing the SbA results.

This research was supported by the Natural Sciences and Engineering Research Council of Canada and the Alberta Informatics Circle of Research Excellence.

## References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Hidden Markov support vector ma-

- chines. *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 3–10.
- Susan Bartlett. 2007. Discriminative approach to automatic syllabification. Master's thesis, Department of Computing Science, University of Alberta.
- Gosse Bouma. 2002. Finite state methods for hyphenation. *Natural Language Engineering*, 1:1–16.
- Stanley Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. *Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech)*.
- Walter Daelemans and Antal van den Bosch. 1992. Generalization performance of backpropagation learning on a syllabification task. *Proceedings of the 3rd Twente Workshop on Language Technology*, pages 27–38.
- Walter Daelemans, Antal van den Bosch, and Ton Weijters. 1997. IGTtree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, pages 407–423.
- Vera Demberg, Helmut Schmid, and Gregor Möhler. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*.
- Vera Demberg. 2006. Letter-to-phoneme conversion for a German text-to-speech system. Master's thesis, University of Stuttgart.
- Philip Babcock Gove, editor. 1993. *Webster's Third New International Dictionary of the English Language, Unabridged*. Merriam-Webster Inc.
- Sittichai Jiampojarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden Markov models to letter-to-phoneme conversion. *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics HLT-NAACL*, pages 372–379.
- Brigitte Krenn. 1997. Tagging syllables. *Proceedings of Eurospeech*, pages 991–994.
- Yannick Marchand and Robert Damper. 2007. Can syllabification improve pronunciation by analogy of English? *Natural Language Engineering*, 13(1):1–24.
- Yannick Marchand, Connie Adsett, and Robert Damper. 2007. Evaluation of automatic syllabification algorithms for English. In *Proceedings of the 6th International Speech Communication Association (ISCA) Workshop on Speech Synthesis*.
- Yannick Marchand. 2007. Personal correspondence.
- Karin Müller. 2001. Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. *Proceedings on the 39th Meeting of the Association for Computational Linguistics (ACL)*, pages 410–417.
- Steve Pearson, Roland Kuhn, Steven Fincke, and Nick Kibre. 2000. Automatic methods for lexical stress assignment and syllabification. In *Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP)*, pages 423–426.
- Helmut Schmid, Bernd Möbius, and Julia Weidenkaff. 2007. Tagging syllable boundaries with joint N-gram models. *Proceedings of Interspeech*.
- Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 823–830.
- Antal van den Bosch. 1997. *Learning to pronounce written words: a study in inductive language learning*. Ph.D. thesis, Universiteit Maastricht.

# A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model

**Libin Shen**

BBN Technologies  
Cambridge, MA 02138, USA  
lshen@bbn.com

**Jinxi Xu**

BBN Technologies  
Cambridge, MA 02138, USA  
jxu@bbn.com

**Ralph Weischedel**

BBN Technologies  
Cambridge, MA 02138, USA  
weisched@bbn.com

## Abstract

In this paper, we propose a novel string-to-dependency algorithm for statistical machine translation. With this new framework, we employ a target dependency language model during decoding to exploit long distance word relations, which are unavailable with a traditional n-gram language model. Our experiments show that the string-to-dependency decoder achieves 1.48 point improvement in BLEU and 2.53 point improvement in TER compared to a standard hierarchical string-to-string system on the NIST 04 Chinese-English evaluation set.

## 1 Introduction

In recent years, hierarchical methods have been successfully applied to Statistical Machine Translation (Graehl and Knight, 2004; Chiang, 2005; Ding and Palmer, 2005; Quirk et al., 2005). In some language pairs, i.e. Chinese-to-English translation, state-of-the-art hierarchical systems show significant advantage over phrasal systems in MT accuracy. For example, Chiang (2007) showed that the Hiero system achieved about 1 to 3 point improvement in BLEU on the NIST 03/04/05 Chinese-English evaluation sets compared to a start-of-the-art phrasal system.

Our work extends the hierarchical MT approach. We propose a string-to-dependency model for MT, which employs rules that represent the source side as strings and the target side as dependency structures. We restrict the target side to the so called *well-formed* dependency structures, in order to cover a large set of non-constituent transfer rules (Marcu et al., 2006), and enable efficient decoding through dynamic programming. We incorporate a dependency

language model during decoding, in order to exploit long-distance word relations which are unavailable with a traditional n-gram language model on target strings.

For comparison purposes, we replicated the Hiero decoder (Chiang, 2005) as our baseline. Our string-to-dependency decoder shows 1.48 point improvement in BLEU and 2.53 point improvement in TER on the NIST 04 Chinese-English MT evaluation set.

In the rest of this section, we will briefly discuss previous work on hierarchical MT and dependency representations, which motivated our research. In section 2, we introduce the model of string-to-dependency decoding. Section 3 illustrates of the use of dependency language models. In section 4, we describe the implementation details of our MT system. We discuss experimental results in section 5, compare to related work in section 6, and draw conclusions in section 7.

### 1.1 Hierarchical Machine Translation

Graehl and Knight (2004) proposed the use of target-tree-to-source-string transducers ( $xRS$ ) to model translation. In  $xRS$  rules, the right-hand-side( $rhs$ ) of the target side is a tree with non-terminals(NTs), while the  $rhs$  of the source side is a string with NTs. Galley et al. (2006) extended this string-to-tree model by using Context-Free parse trees to represent the target side. A tree could represent multi-level transfer rules.

The Hiero decoder (Chiang, 2007) does not require explicit syntactic representation on either side of the rules. Both source and target are strings with NTs. Decoding is solved as chart parsing. Hiero can be viewed as a hierarchical string-to-string model.

Ding and Palmer (2005) and Quirk et al. (2005)

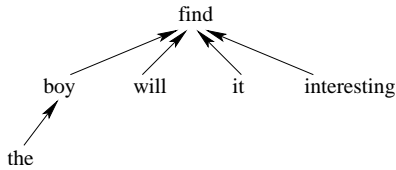


Figure 1: The dependency tree for sentence *the boy will find it interesting*

followed the tree-to-tree approach (Shieber and Schabes, 1990) for translation. In their models, dependency treelets are used to represent both the source and the target sides. Decoding is implemented as tree transduction preceded by source side dependency parsing. While tree-to-tree models can represent richer structural information, existing tree-to-tree models did not show advantage over string-to-tree models on translation accuracy due to a much larger search space.

One of the motivations of our work is to achieve desirable trade-off between model capability and search space through the use of the so called *well-formed* dependency structures in rule representation.

## 1.2 Dependency Trees

Dependency trees reveal long-distance relations between words. For a given sentence, each word has a parent word which it depends on, except for the *root* word.

Figure 1 shows an example of a dependency tree. Arrows point from the child to the parent. In this example, the word *find* is the root.

Dependency trees are simpler in form than CFG trees since there are no constituent labels. However, dependency relations directly model semantic structure of a sentence. As such, dependency trees are a desirable prior model of the target sentence.

## 1.3 Motivations for Well-Formed Dependency Structures

We restrict ourselves to the so-called *well-formed* target dependency structures based on the following considerations.

### Dynamic Programming

In (Ding and Palmer, 2005; Quirk et al., 2005), there is no restriction on dependency treelets used in transfer rules except for the size limit. This may result in a high dimensionality in hypothesis represen-

tation and make it hard to employ shared structures for efficient dynamic programming.

In (Galley et al., 2004), rules contain NT slots and combination is only allowed at those slots. Therefore, the search space becomes much smaller. Furthermore, shared structures can be easily defined based on the labels of the slots.

In order to take advantage of dynamic programming, we fixed the positions onto which another another tree could be attached by specifying NTs in dependency trees.

### Rule Coverage

Marcu et al. (2006) showed that many useful phrasal rules cannot be represented as hierarchical rules with the existing representation methods, even with composed transfer rules (Galley et al., 2006). For example, the following rule

- $\langle (\text{hong})_{\text{Chinese}}, (\text{DT}(\text{the}) \text{JJ}(\text{red}))_{\text{English}} \rangle$

is not a valid string-to-tree transfer rule since *the red* is a partial constituent.

A number of techniques have been proposed to improve rule coverage. (Marcu et al., 2006) and (Galley et al., 2006) introduced artificial constituent nodes dominating the phrase of interest. The binarization method used by Wang et al. (2007) can cover many non-constituent rules also, but not all of them. For example, it cannot handle the above example. DeNeefe et al. (2007) showed that the best results were obtained by combing these methods.

In this paper, we use *well-formed* dependency structures to handle the coverage of non-constituent rules. The use of dependency structures is due to the flexibility of dependency trees as a representation method which does not rely on constituents (Fox, 2002; Ding and Palmer, 2005; Quirk et al., 2005). The *well-formedness* of the dependency structures enables efficient decoding through dynamic programming.

## 2 String-to-Dependency Translation

### 2.1 Transfer Rules with Well-Formed Dependency Structures

A string-to-dependency grammar  $G$  is a 4-tuple  $G = \langle \mathcal{R}, X, T_f, T_e \rangle$ , where  $\mathcal{R}$  is a set of transfer rules.  $X$  is the only non-terminal, which is similar to the Hiero system (Chiang, 2007).  $T_f$  is a set of

terminals in the source language, and  $T_e$  is a set of terminals in the target language<sup>1</sup>.

A string-to-dependency transfer rule  $R \in \mathcal{R}$  is a 4-tuple  $R = \langle S_f, S_e, D, A \rangle$ , where  $S_f \in (T_f \cup \{X\})^+$  is a source string,  $S_e \in (T_e \cup \{X\})^+$  is a target string,  $D$  represents the dependency structure for  $S_e$ , and  $A$  is the alignment between  $S_f$  and  $S_e$ . Non-terminal alignments in  $A$  must be one-to-one.

In order to exclude undesirable structures, we only allow  $S_e$  whose dependency structure  $D$  is *well-formed*, which we will define below. In addition, the same well-formedness requirement will be applied to partial decoding results. Thus, we will be able to employ shared structures to merge multiple partial results.

Based on the results in previous work (DeNeeffe et al., 2007), we want to keep two kinds of dependency structures. In one kind, we keep dependency trees with a sub-root, where all the children of the sub-root are complete. We call them **fixed** dependency structures because the head is known or fixed. In the other, we keep dependency structures of sibling nodes of a common head, but the head itself is unspecified or floating. Each of the siblings must be a complete constituent. We call them **floating** dependency structures. Floating structures can represent many linguistically meaningful non-constituent structures: for example, like *the red*, a modifier of a noun. Only those two kinds of dependency structures are **well-formed** structures in our system.

Furthermore, we operate over *well-formed* structures in a bottom-up style in decoding. However, the description given above does not provide a clear definition on how to combine those two types of structures. In the rest of this section, we will provide formal definitions of *well-formed* structures and combinatory operations over them, so that we can easily manipulate *well-formed* structures in decoding. Formal definitions also allow us to easily extend the framework to incorporate a dependency language model in decoding. Examples will be provided along with the formal definitions.

Consider a sentence  $S = w_1 w_2 \dots w_n$ . Let  $d_1 d_2 \dots d_n$  represent the parent word IDs for each word. For example,  $d_4 = 2$  means that  $w_4$  depends

<sup>1</sup>We ignore the left hand side here because there is only one non-terminal  $X$ . Of course, this formalism can be extended to have multiple NTs.

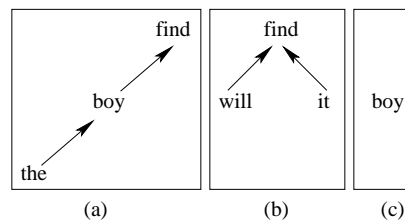


Figure 2: Fixed dependency structures

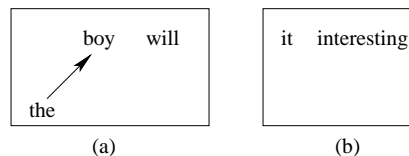


Figure 3: Floating dependency structures

on  $w_2$ . If  $w_i$  is a root, we define  $d_i = 0$ .

**Definition 1** A dependency structure  $d_{i..j}$  is **fixed on head  $h$** , where  $h \in [i, j]$ , or **fixed for short**, if and only if it meets the following conditions

- $d_h \notin [i, j]$
- $\forall k \in [i, j]$  and  $k \neq h$ ,  $d_k \in [i, j]$
- $\forall k \notin [i, j]$ ,  $d_k = h$  or  $d_k \notin [i, j]$

In addition, we say the category of  $d_{i..j}$  is  $(-, h, -)$ , where  $-$  means this field is undefined.

**Definition 2** A dependency structure  $d_{i..j}$  is **floating with children  $C$** , for a non-empty set  $C \subseteq \{i, \dots, j\}$ , or **floating for short**, if and only if it meets the following conditions

- $\exists h \notin [i, j]$ , s.t.  $\forall k \in C$ ,  $d_k = h$
- $\forall k \in [i, j]$  and  $k \notin C$ ,  $d_k \in [i, j]$
- $\forall k \notin [i, j]$ ,  $d_k \notin [i, j]$

We say the category of  $d_{i..j}$  is  $(C, -, -)$  if  $j < h$ , or  $(-, -, C)$  otherwise. A category is composed of the three fields  $(A, h, B)$ , where  $h$  is used to represent the head, and  $A$  and  $B$  are designed to model left and right dependents of the head respectively.

A dependency structure is **well-formed** if and only if it is either *fixed* or *floating*.

### Examples

We can represent dependency structures with graphs. Figure 2 shows examples of fixed structures, Figure 3 shows examples of floating structures, and Figure 4 shows ill-formed dependency structures.

It is easy to verify that the structures in Figures 2 and 3 are well-formed. 4(a) is ill-formed because

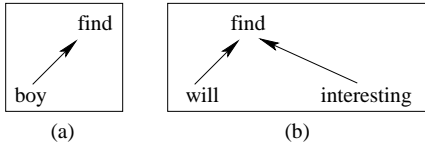


Figure 4: Ill-formed dependency structures

*boy* does not have its child word *the* in the tree. 4(b) is ill-formed because it is not a continuous segment.

As for the example *the red* mentioned above, it is a *well-formed* floating dependency structure.

## 2.2 Operations on Well-Formed Dependency Structures and Categories

One of the purposes of introducing *floating* dependency structures is that siblings having a common parent will become a well-defined entity, although they are not considered a constituent. We always build well-formed partial structures on the target side in decoding. Furthermore, we combine partial dependency structures in a way such that we can obtain all possible well-formed but no ill-formed dependency structures during bottom-up decoding.

The solution is to employ *categories* introduced above. Each well-formed dependency structure has a category. We can apply four combinatory operations over the categories. If we can combine two categories with a certain *category operation*, we can use a corresponding *tree operation* to combine two dependency structures. The category of the combined dependency structure is the result of the combinatory *category operations*.

We first introduce three meta category operations. Two of them are unary operations, *left raising* (LR) and *right raising* (RR), and one is the binary operation *unification* (UF).

First, the raising operations are used to turn a completed fixed structure into a floating structure. It is easy to verify the following theorem according to the definitions.

**Theorem 1** *A fixed structure with category  $(-, h, -)$  for span  $[i, j]$  is also a floating structure with children  $\{h\}$  if there are no outside words depending on word  $h$ .*

$$\forall k \notin [i, j], d_k \neq h. \quad (1)$$

Therefore we can always *raise* a fixed structure if we assume it is complete, i.e. (1) holds.

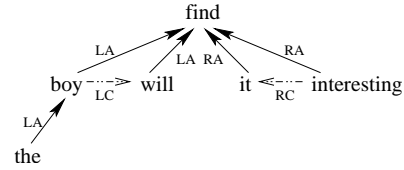


Figure 5: A dependency tree with flexible combination

### Definition 3 Meta Category Operations

- $LR((-, h, -)) = (\{h\}, -, -)$
- $RR((-, h, -)) = (-, -, \{h\})$
- $UF((A_1, h_1, B_1), (A_2, h_2, B_2)) = \text{NORM}((A_1 \sqcup A_2, h_1 \sqcup h_2, B_1 \sqcup B_2))$

Unification is well-defined if and only if we can unify all three elements and the result is a valid fixed or floating category. For example, we can unify a fixed structure with a floating structure or two floating structures in the same direction, but we cannot unify two fixed structures.

$$h_1 \sqcup h_2 = \begin{cases} h_1 & \text{if } h_2 = - \\ h_2 & \text{if } h_1 = - \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$A_1 \sqcup A_2 = \begin{cases} A_1 & \text{if } A_2 = - \\ A_2 & \text{if } A_1 = - \\ A_1 \cup A_2 & \text{otherwise} \end{cases}$$

$$\text{NORM}((A, h, B)) = \begin{cases} (-, h, -) & \text{if } h \neq - \\ (A, -, -) & \text{if } h = -, B = - \\ (-, -, B) & \text{if } h = -, A = - \\ \text{undefined} & \text{otherwise} \end{cases}$$

Next we introduce the four tree operations on *dependency structures*. Instead of providing the formal definition, we use figures to illustrate these operations to make it easy to understand. Figure 1 shows a traditional dependency tree. Figure 5 shows the four operations to combine partial dependency structures, which are *left adjoining* (LA), *right adjoining* (RA), *left concatenation* (LC) and *right concatenation* (RC).

Child and parent subtrees can be combined with *adjoining* which is similar to the traditional dependency formalism. We can either adjoin a fixed structure or a floating structure to the head of a fixed structure.

Complete siblings can be combined via *concatenation*. We can concatenate two fixed structures, one fixed structure with one floating structure, or two floating structures in the same direction. The flexibility of the order of operation allows us to take ad-



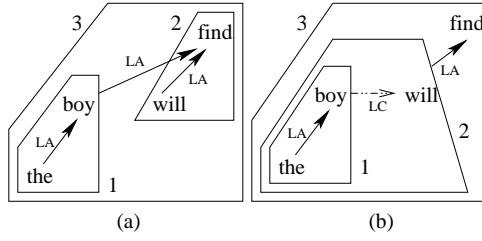


Figure 6: Operations over well-formed structures

vantage of various translation fragments encoded in transfer rules.

Figure 6 shows alternative ways of applying operations on *well-formed* structures to build larger structures in a bottom-up style. Numbers represent the order of operation.

We use the same names for the operations on categories for the sake of convenience. We can easily use the meta category operations to define the four combinatory operations. The definition of the operations in the left direction is as follows. Those in the right direction are similar.

**Definition 4** *Combinatory category operations*

$$\begin{aligned}
& \text{LA}((A_1, -, -), (-, h_2, -)) \\
= & \text{UF}((A_1, -, -), (-, h_2, -)) \\
& \text{LA}((-, h_1, -), (-, h_2, -)) \\
= & \text{UF}(\text{LR}((-, h_1, -)), (-, h_2, -)) \\
& \text{LC}((A_1, -, -), (A_2, -, -)) \\
= & \text{UF}((A_1, -, -), (A_2, -, -)) \\
& \text{LC}((A_1, -, -), (-, h_2, -)) \\
= & \text{UF}((A_1, -, -), \text{LR}((-, h_2, -))) \\
& \text{LC}((-, h_1, -), (A_2, -, -)) \\
= & \text{UF}(\text{LR}((-, h_1, -)), (A_2, -, -)) \\
& \text{LC}((-, h_1, -), (-, h_2, -)) \\
= & \text{UF}(\text{LR}((-, h_1, -)), \text{LR}((-, h_2, -)))
\end{aligned}$$

It is easy to verify the *soundness* and *completeness* of category operations based on one-to-one mapping of the conditions in the definitions of corresponding operations on dependency structures and on categories.

**Theorem 2 (soundness and completeness)**

Suppose  $X$  and  $Y$  are well-formed dependency structures.  $OP(\text{cat}(X), \text{cat}(Y))$  is well-defined for a given operation  $OP$  if and only if  $OP(X, Y)$  is well-defined. Furthermore,

$$\text{cat}(OP(X, Y)) = OP(\text{cat}(X), \text{cat}(Y))$$

Suppose we have a dependency tree for *a red apple*, where both  $a$  and  $red$  depend on  $apple$ . There are two ways to compute the category of this string from the bottom up.

$$\begin{aligned}
& \text{cat}(D_{a\_red\_apple}) \\
= & \text{LA}(\text{cat}(D_a), \text{LA}(\text{cat}(D_{red}), \text{cat}(D_{apple}))) \\
= & \text{LA}(\text{LC}(\text{cat}(D_a), \text{cat}(D_{red})), \text{cat}(D_{apple}))
\end{aligned}$$

Based on Theorem 2, it follows that combinatory operation of categories has the *confluence* property, since the result dependency structure is determined.

**Corollary 1 (confluence)** *The category of a well-formed dependency tree does not depend on the order of category calculation.*

With categories, we can easily track the types of dependency structures and constrain operations in decoding. For example, we have a rule with dependency structure  $find \leftarrow X$ , where  $X$  right adjoins to  $find$ . Suppose we have two floating structures<sup>2</sup>,

$$\begin{aligned}
\text{cat}(X_1) &= (\{he, will\}, -, -) \\
\text{cat}(X_2) &= (-, -, \{it, interesting\})
\end{aligned}$$

We can replace  $X$  by  $X_2$ , but not by  $X_1$  based on the definition of category operations.

**2.3 Rule Extraction**

Now we explain how we get the string-to-dependency rules from training data. The procedure is similar to (Chiang, 2007) except that we maintain tree structures on the target side, instead of strings.

Given sentence-aligned bi-lingual training data, we first use GIZA++ (Och and Ney, 2003) to generate word level alignment. We use a statistical CFG parser to parse the English side of the training data, and extract dependency trees with Magerman’s rules (1995). Then we use heuristic rules to extract transfer rules recursively based on the GIZA alignment and the target dependency trees. The rule extraction procedure is as follows.

1. Initialization:

All the 4-tuples  $(P_f^{i,j}, P_e^{m,n}, D, A)$  are valid phrase alignments, where source phrase  $P_f^{i,j}$  is

<sup>2</sup>Here we use words instead of word indexes in categories to make the example easy to understand.

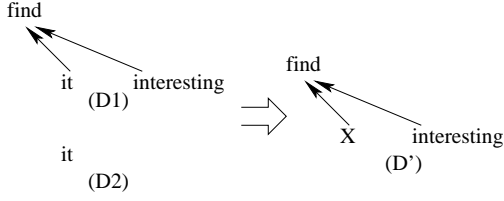


Figure 7: Replacing *it* with *X* in  $D_1$

aligned to target phrase  $P_e^{m,n}$  under alignment<sup>3</sup>  $A$ , and  $D$ , the dependency structure for  $P_e^{m,n}$ , is *well-formed*. All *valid phrase templates* are *valid rules templates*.

## 2. Inference:

Let  $(P_f^{i,j}, P_e^{m,n}, D_1, A)$  be a *valid rule template*, and  $(P_f^{p,q}, P_e^{s,t}, D_2, A)$  a *valid phrase alignment*, where  $[p, q] \subset [i, j]$ ,  $[s, t] \subset [m, n]$ ,  $D_2$  is a sub-structure of  $D_1$ , and at least one word in  $P_f^{i,j}$  but not in  $P_f^{p,q}$  is aligned.

We create a new *valid rule template*  $(P'_f, P'_e, D', A)$ , where we obtain  $P'_f$  by replacing  $P_f^{p,q}$  with label  $X$  in  $P_f^{i,j}$ , and obtain  $P'_e$  by replacing  $P_e^{s,t}$  with  $X$  in  $P_e^{m,n}$ . Furthermore, We obtain  $D'$  by replacing sub-structure  $D_2$  with  $X$  in  $D_1$ <sup>4</sup>. An example is shown in Figure 7.

Among all valid rule templates, we collect those that contain at most two NTs and at most seven elements in the source as transfer rules in our system.

## 2.4 Decoding

Following previous work on hierarchical MT (Chiang, 2005; Galley et al., 2006), we solve decoding as chart parsing. We view target dependency as the hidden structure of source fragments.

The parser scans all source cells in a bottom-up style, and checks matched transfer rules according to the source side. Once there is a completed rule, we build a larger dependency structure by substituting component dependency structures for corresponding NTs in the target dependency structure of rules.

Hypothesis dependency structures are organized in a shared forest, or AND-OR structures. An AND-

<sup>3</sup>By  $P_f^{i,j}$  aligned to  $P_e^{m,n}$ , we mean all words in  $P_f^{i,j}$  are either aligned to words in  $P_e^{m,n}$  or unaligned, and vice versa. Furthermore, at least one word in  $P_f^{i,j}$  is aligned to a word in  $P_e^{m,n}$ .

<sup>4</sup>If  $D_2$  is a *floating* structure, we need to merge several dependency links into one.

structure represents an application of a rule over component OR-structures, and an OR-structure represents a set of alternative AND-structures with the same *state*. A **state** means a  $n$ -tuple that characterizes the information that will be inquired by up-level AND-structures.

Supposing we use a traditional tri-gram language model in decoding, we need to specify the leftmost two words and the rightmost two words in a state. Since we only have a single NT  $X$  in the formalism described above, we do not need to add the NT label in states. However, we need to specify one of the three types of the dependency structure: fixed, floating on the left side, or floating on the right side. This information is encoded in the category of the dependency structure.

In the next section, we will explain how to extend categories and states to exploit a dependency language model during decoding.

## 3 Dependency Language Model

For the dependency tree in Figure 1, we calculate the probability of the tree as follows

$$\begin{aligned}
 Prob &= P_T(find) \\
 &\times P_L(will|find-as-head) \\
 &\times P_L(boy|will, find-as-head) \\
 &\times P_L(the|boy-as-head) \\
 &\times P_R(it|find-as-head) \\
 &\times P_R(interesting|it, find-as-head)
 \end{aligned}$$

Here  $P_T(x)$  is the probability that word  $x$  is the root of a dependency tree.  $P_L$  and  $P_R$  are left and right side generative probabilities respectively. Let  $w_h$  be the head, and  $w_{L_1}w_{L_2}\dots w_{L_n}$  be the children on the left side from the nearest to the farthest. Suppose we use a tri-gram dependency LM,

$$\begin{aligned}
 &P_L(w_{L_1}w_{L_2}\dots w_{L_n}|w_h-as-head) \\
 = &P_L(w_{L_1}|w_h-as-head) \\
 &\times P_L(w_{L_2}|w_{L_1}, w_h-as-head) \\
 &\times \dots \times P_L(w_{L_n}|w_{L_{n-1}}, w_{L_{n-2}}) \quad (2)
 \end{aligned}$$

$w_h$ -as-head represents  $w_h$  used as the head, and it is different from  $w_h$  in the dependency language model. The right side probability is similar.

In order to calculate the dependency language model score, or depLM score for short, on the fly for

partial hypotheses in a bottom-up decoding, we need to save more information in categories and states.

We use a 5-tuple  $(LF, LN, h, RN, RF)$  to represent the category of a dependency structure.  $h$  represents the head.  $LF$  and  $RF$  represent the farthest two children on the left and right sides respectively. Similarly,  $LN$  and  $RN$  represent the nearest two children on the left and right sides respectively. The three types of categories are as follows.

- fixed:  $(LF, -, h, -, RF)$
- floating left:  $(LF, LN, -, -, -)$
- floating right:  $(-, -, -, RN, RF)$

Similar operations as described in Section 2.2 are used to keep track of the head and boundary child nodes which are then used to compute depLM scores in decoding. Due to the limit of space, we skip the details here.

## 4 Implementation Details

### Features

1. Probability of the source side given the target side of a rule
2. Probability of the target side given the source side of a rule
3. Word alignment probability
4. Number of target words
5. Number of concatenation rules used
6. Language model score
7. Dependency language model score
8. Discount on ill-formed dependency structures

We have eight features in our system. The values of the first four features are accumulated on the rules used in a translation. Following (Chiang, 2005), we also use concatenation rules like  $X \rightarrow XX$  for backup. The 5th feature counts the number of concatenation rules used in a translation. In our system, we allow substitutions of dependency structures with unmatched categories, but there is a discount for such substitutions.

### Weight Optimization

We tune the weights with several rounds of decoding-optimization. Following (Och, 2003), the k-best results are accumulated as the input of the optimizer. Powell’s method is used for optimization with 20 random starting points around the weight vector of the last iteration.

## Rescoring

We rescore 1000-best translations (Huang and Chiang, 2005) by replacing the 3-gram LM score with the 5-gram LM score computed offline.

## 5 Experiments

We carried out experiments on three models.

- baseline: replication of the Hiero system.
- filtered: a string-to-string MT system as in baseline. However, we only keep the transfer rules whose target side can be generated by a well-formed dependency structure.
- str-dep: a string-to-dependency system with a dependency LM.

We take the replicated Hiero system as our baseline because it is the closest to our string-to-dependency model. They have similar rule extraction and decoding algorithms. Both systems use only one non-terminal label in rules. The major difference is in the representation of target structures. We use dependency structures instead of strings; thus, the comparison will show the contribution of using dependency information in decoding.

All models are tuned on BLEU (Papineni et al., 2001), and evaluated on both BLEU and Translation Error Rate (TER) (Snover et al., 2006) so that we could detect over-tuning on one metric.

We used part of the NIST 2006 Chinese-English large track data as well as some LDC corpora collected for the DARPA GALE program (LDC2005E83, LDC2006E34 and LDC2006G05) as our bilingual training data. It contains about 178M/191M words in source/target. Hierarchical rules were extracted from a subset which has about 35M/41M words<sup>5</sup>, and the rest of the training data were used to extract phrasal rules as in (Och, 2003; Chiang, 2005). The English side of this subset was also used to train a 3-gram dependency LM. Traditional 3-gram and 5-gram LMs were trained on a corpus of 6G words composed of the LDC Gigaword corpus and text downloaded from Web (Bulyko et al., 2007). We tuned the weights on NIST MT05 and tested on MT04.

<sup>5</sup>It includes eight corpora: LDC2002E18, LDC2003E07, LDC2004T08\_HK\_News, LDC2005E83, LDC2005T06, LDC2005T10, LDC2006E34, and LDC2006G05

Model	#Rules
baseline	140M
filtered	26M
str-dep	27M

Table 1: Number of transfer rules

Model	BLEU%		TER%	
	lower	mixed	lower	mixed
Decoding (3-gram LM)				
baseline	38.18	35.77	58.91	56.60
filtered	37.92	35.48	57.80	55.43
str-dep	39.52	37.25	56.27	54.07
Rescoring (5-gram LM)				
baseline	40.53	38.26	56.35	54.15
filtered	40.49	38.26	55.57	53.47
str-dep	41.60	39.47	55.06	52.96

Table 2: BLEU and TER scores on the test set.

Table 1 shows the number of transfer rules extracted from the training data for the tuning and test sets. The constraint of well-formed dependency structures greatly reduced the size of the rule set. Although the rule size increased a little bit after incorporating dependency structures in rules, the size of string-to-dependency rule set is less than 20% of the baseline rule set size.

Table 2 shows the BLEU and TER scores on MT04. On decoding output, the string-to-dependency system achieved 1.48 point improvement in BLEU and 2.53 point improvement in TER compared to the baseline hierarchical string-to-string system. After 5-gram rescoring, it achieved 1.21 point improvement in BLEU and 1.19 improvement in TER. The *filtered* model does not show improvement on BLEU. The filtered string-to-string rules can be viewed the string projection of string-to-dependency rules. It means that just using dependency structure does not provide an improvement on performance. However, dependency structures allow the use of a dependency LM which gives rise to significant improvement.

## 6 Discussion

The *well-formed* dependency structures defined here are similar to the data structures in previous work on mono-lingual parsing (Eisner and Satta, 1999; McDonald et al., 2005). However, here we have *fixed* structures growing on both sides to exploit various translation fragments learned in the training data,

while the operations in mono-lingual parsing were designed to avoid artificial ambiguity of derivation.

Charniak et al. (2003) described a two-step string-to-CFG-tree translation model which employed a syntax-based language model to select the best translation from a target parse forest built in the first step. Only translation probability  $P(F|E)$  was employed in the construction of the target forest due to the complexity of the syntax-based LM. Since our dependency LM models structures over target words directly based on dependency trees, we can build a single-step system. This dependency LM can also be used in hierarchical MT systems using lexicalized CFG trees.

The use of a dependency LM in MT is similar to the use of a structured LM in ASR (Xu et al., 2002), which was also designed to exploit long-distance relations. The depLM is used in a bottom-up style, while SLM is employed in a left-to-right style.

## 7 Conclusions and Future Work

In this paper, we propose a novel string-to-dependency algorithm for statistical machine translation. For comparison purposes, we replicated the Hiero system as described in (Chiang, 2005). Our string-to-dependency system generates 80% fewer rules, and achieves 1.48 point improvement in BLEU and 2.53 point improvement in TER on the decoding output on the NIST 04 Chinese-English evaluation set.

Dependency structures provide a desirable platform to employ linguistic knowledge in MT. In the future, we will continue our research in this direction to carry out translation with deeper features, for example, propositional structures (Palmer et al., 2005). We believe that the *fixed* and *floating* structures proposed in this paper can be extended to model predicates and arguments.

## Acknowledgments

This work was supported by DARPA/IPTO Contract No. HR0011-06-C-0022 under the GALE program. We are grateful to Roger Bock, Ivan Bulyko, Mike Kayser, John Makhoul, Spyros Matsoukas, Antti-Veikko Rosti, Rich Schwartz and Bing Zhang for their help in running the experiments and constructive comments to improve this paper.

## References

- I. Bulyko, S. Matsoukas, R. Schwartz, L. Nguyen, and J. Makhoul. 2007. Language model adaptation in machine translation from speech. In *Proceedings of the 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.
- E. Charniak, K. Knight, and K. Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit IX*.
- D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*.
- D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).
- S. DeNeefe, K. Knight, W. Wang, and D. Marcu. 2007. What can syntax-based mt learn from phrase-based mt? In *Proceedings of the 2007 Conference of Empirical Methods in Natural Language Processing*.
- Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 541–548, Ann Arbor, Michigan, June.
- J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*.
- H. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of the 2002 Conference of Empirical Methods in Natural Language Processing*.
- M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a translation rule? In *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.
- M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable inference and training of context-rich syntactic models. In *COLING-ACL '06: Proceedings of 44th Annual Meeting of the Association for Computational Linguistics and 21st Int. Conf. on Computational Linguistics*.
- J. Graehl and K. Knight. 2004. Training tree transducers. In *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.
- L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies*.
- D. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.
- D. Marcu, W. Wang, A. Echihiabi, and K. Knight. 2006. SPMT: Statistical machine translation with syntactically defined target language phrases. In *Proceedings of the 2006 Conference of Empirical Methods in Natural Language Processing*.
- R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*.
- F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).
- F. J. Och. 2003. Minimum error rate training for statistical machine translation. In Erhard W. Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.
- M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).
- K. Papineni, S. Roukos, and T. Ward. 2001. Bleu: a method for automatic evaluation of machine translation. IBM Research Report, RC22176.
- C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–279, Ann Arbor, Michigan, June.
- S. Shieber and Y. Schabes. 1990. Synchronous tree adjoining grammars. In *Proceedings of COLING '90: The 13th Int. Conf. on Computational Linguistics*.
- M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.
- W. Wang, K. Knight, and D. Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Conference of Empirical Methods in Natural Language Processing*.
- P. Xu, C. Chelba, and F. Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

# Forest Reranking: Discriminative Parsing with Non-Local Features\*

Liang Huang

University of Pennsylvania  
Philadelphia, PA 19104  
lh Huang3@cis.upenn.edu

## Abstract

Conventional  $n$ -best reranking techniques often suffer from the limited scope of the  $n$ -best list, which rules out many potentially good alternatives. We instead propose *forest reranking*, a method that reranks a packed forest of exponentially many parses. Since exact inference is intractable with non-local features, we present an approximate algorithm inspired by forest rescoring that makes discriminative training practical over the whole Treebank. Our final result, an F-score of 91.7, outperforms both 50-best and 100-best reranking baselines, and is better than any previously reported systems trained on the Treebank.

## 1 Introduction

Discriminative reranking has become a popular technique for many NLP problems, in particular, parsing (Collins, 2000) and machine translation (Shen et al., 2005). Typically, this method first generates a list of top- $n$  candidates from a baseline system, and then reranks this  $n$ -best list with arbitrary features that are not computable or intractable to compute within the baseline system. But despite its apparent success, there remains a major drawback: this method suffers from the limited scope of the  $n$ -best list, which rules out many potentially good alternatives. For example 41% of the correct parses were not in the candidates of  $\sim 30$ -best parses in (Collins, 2000). This situation becomes worse with longer sentences because the number of possible interpretations usually grows exponentially with the

\* Part of this work was done while I was visiting Institute of Computing Technology, Beijing, and I thank Prof. Qun Liu and his lab for hosting me. I am also grateful to Dan Gildea and Mark Johnson for inspirations, Eugene Charniak for help with his parser, and Wenbin Jiang for guidance on perceptron averaging. This project was supported by NSF ITR EIA-0205456.

	<i>local</i>	<i>non-local</i>
conventional reranking	only at the root	
DP-based discrim. parsing	exact	N/A
<i>this work: forest-reranking</i>	exact	<i>on-the-fly</i>

Table 1: Comparison of various approaches for incorporating local and non-local features.

sentence length. As a result, we often see very few variations among the  $n$ -best trees, for example, 50-best trees typically just represent a combination of 5 to 6 binary ambiguities (since  $2^5 < 50 < 2^6$ ).

Alternatively, discriminative parsing is tractable with exact and efficient search based on dynamic programming (DP) if all features are restricted to be *local*, that is, only looking at a local window within the factored search space (Taskar et al., 2004; McDonald et al., 2005). However, we miss the benefits of non-local features that are not representable here.

Ideally, we would wish to combine the merits of both approaches, where an efficient inference algorithm could integrate both local and non-local features. Unfortunately, exact search is intractable (at least in theory) for features with unbounded scope. So we propose *forest reranking*, a technique inspired by forest rescoring (Huang and Chiang, 2007) that approximately reranks the packed forest of exponentially many parses. The key idea is to compute non-local features incrementally from bottom up, so that we can rerank the  $n$ -best subtrees at all internal nodes, instead of only at the root node as in conventional reranking (see Table 1). This method can thus be viewed as a step towards the integration of discriminative reranking with traditional chart parsing.

Although previous work on discriminative parsing has mainly focused on short sentences ( $\leq 15$  words) (Taskar et al., 2004; Turian and Melamed, 2007), our work scales to the whole Treebank, where

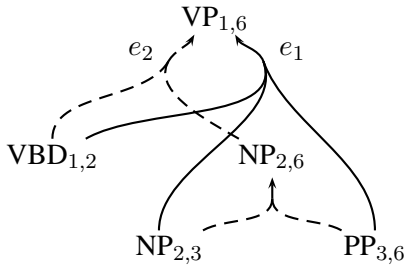


Figure 1: A partial forest of the example sentence.

we achieved an F-score of 91.7, which is a 19% error reduction from the 1-best baseline, and outperforms both 50-best and 100-best reranking. This result is also better than any previously reported systems trained on the Treebank.

## 2 Packed Forests as Hypergraphs

Informally, a packed parse forest, or *forest* in short, is a compact representation of all the derivations (i.e., parse trees) for a given sentence under a context-free grammar (Billot and Lang, 1989). For example, consider the following sentence

<sub>0</sub> I <sub>1</sub> saw <sub>2</sub> him <sub>3</sub> with <sub>4</sub> a <sub>5</sub> mirror <sub>6</sub>

where the numbers between words denote string positions. Shown in Figure 1, this sentence has (at least) two derivations depending on the attachment of the prep. phrase  $PP_{3,6}$  “with a mirror”: it can either be attached to the verb “saw”,

$$\frac{VBD_{1,2} \quad NP_{2,3} \quad PP_{3,6}}{VP_{1,6}}, \quad (*)$$

or be attached to “him”, which will be further combined with the verb to form the same VP as above. These two derivations can be represented as a single forest by sharing common sub-derivations. Such a forest has a structure of a hypergraph (Klein and Manning, 2001; Huang and Chiang, 2005), where items like  $PP_{3,6}$  are called *nodes*, and deductive steps like (\*) correspond to *hyperedges*.

More formally, a **forest** is a pair  $\langle V, E \rangle$ , where  $V$  is the set of **nodes**, and  $E$  the set of **hyperedges**. For a given sentence  $w_{1:l} = w_1 \dots w_l$ , each node  $v \in V$  is in the form of  $X_{i,j}$ , which denotes the recognition of nonterminal  $X$  spanning the substring from positions  $i$  through  $j$  (that is,  $w_{i+1} \dots w_j$ ). Each hyperedge  $e \in E$  is a pair  $\langle tails(e), head(e) \rangle$ , where

$head(e) \in V$  is the consequent node in the deductive step, and  $tails(e) \in V^*$  is the list of antecedent nodes. For example, the hyperedge for deduction (\*) is notated:

$$e_1 = \langle (VBD_{1,2}, NP_{2,3}, PP_{3,6}), VP_{1,6} \rangle$$

We also denote  $IN(v)$  to be the set of **incoming hyperedges** of node  $v$ , which represent the different ways of deriving  $v$ . For example, in the forest in Figure 1,  $IN(VP_{1,6})$  is  $\{e_1, e_2\}$ , with  $e_2 = \langle (VBD_{1,2}, NP_{2,6}), VP_{1,6} \rangle$ . We call  $|e|$  the **arity** of hyperedge  $e$ , which counts the number of tail nodes in  $e$ . The arity of a hypergraph is the maximum arity over all hyperedges. A CKY forest has an arity of 2, since the input grammar is required to be binary branching (cf. Chomsky Normal Form) to ensure cubic time parsing complexity. However, in this work, we use forests from a Treebank parser (Charniak, 2000) whose grammar is often flat in many productions. For example, the arity of the forest in Figure 1 is 3. Such a Treebank-style forest is easier to work with for reranking, since many features can be directly expressed in it. There is also a distinguished **root node** TOP in each forest, denoting the goal item in parsing, which is simply  $S_{0,l}$  where  $S$  is the start symbol and  $l$  is the sentence length.

## 3 Forest Reranking

### 3.1 Generic Reranking with the Perceptron

We first establish a unified framework for parse reranking with both  $n$ -best lists and packed forests.

For a given sentence  $s$ , a generic reranker selects the best parse  $\hat{y}$  among the set of candidates  $cand(s)$  according to some scoring function:

$$\hat{y} = \operatorname{argmax}_{y \in cand(s)} score(y) \quad (1)$$

In  $n$ -best reranking,  $cand(s)$  is simply a set of  $n$ -best parses from the baseline parser, that is,  $cand(s) = \{y_1, y_2, \dots, y_n\}$ . Whereas in forest reranking,  $cand(s)$  is a forest implicitly representing the set of exponentially many parses.

As usual, we define the score of a parse  $y$  to be the dot product between a high dimensional feature representation and a weight vector  $\mathbf{w}$ :

$$score(y) = \mathbf{w} \cdot \mathbf{f}(y) \quad (2)$$

where the feature extractor  $\mathbf{f}$  is a vector of  $d$  functions  $\mathbf{f} = (f_1, \dots, f_d)$ , and each feature  $f_j$  maps a parse  $y$  to a real number  $f_j(y)$ . Following (Charniak and Johnson, 2005), the first feature  $f_1(y) = \log \Pr(y)$  is the log probability of a parse from the baseline generative parser, while the remaining features are all integer valued, and each of them counts the number of times that a particular configuration occurs in parse  $y$ . For example, one such feature  $f_{2000}$  might be a question

“how many times is a VP of length 5 surrounded by the word ‘has’ and the period?”

which is an instance of the **WordEdges** feature (see Figure 2(c) and Section 3.2 for details).

Using a machine learning algorithm, the weight vector  $\mathbf{w}$  can be estimated from the training data where each sentence  $s_i$  is labelled with its correct (“**gold-standard**”) parse  $y_i^*$ . As for the learner, Collins (2000) uses the boosting algorithm and Charniak and Johnson (2005) use the maximum entropy estimator. In this work we use the averaged perceptron algorithm (Collins, 2002) since it is an online algorithm much simpler and orders of magnitude faster than Boosting and MaxEnt methods.

Shown in Pseudocode 1, the perceptron algorithm makes several passes over the whole training data, and in each iteration, for each sentence  $s_i$ , it tries to predict a best parse  $\hat{y}_i$  among the candidates  $\text{cand}(s_i)$  using the current weight setting. Intuitively, we want the gold parse  $y_i^*$  to be picked, but in general it is *not* guaranteed to be within  $\text{cand}(s_i)$ , because the grammar may fail to cover the gold parse, and because the gold parse may be pruned away due to the limited scope of  $\text{cand}(s_i)$ . So we define an **oracle parse**  $y_i^+$  to be the candidate that has the highest Parseval F-score with respect to the gold tree  $y_i^*$ :<sup>1</sup>

$$y_i^+ \triangleq \operatorname{argmax}_{y \in \text{cand}(s_i)} F(y, y_i^*) \quad (3)$$

where function  $F$  returns the F-score. Now we train the reranker to pick the oracle parses as often as possible, and in case an error is made (line 6), perform an update on the weight vector (line 7), by adding the difference between two feature representations.

<sup>1</sup>If one uses the gold  $y_i^*$  for oracle  $y_i^+$ , the perceptron will continue to make updates towards something unreachable even when the decoder has picked the best possible candidate.

---

### Pseudocode 1 Perceptron for Generic Reranking

---

```

1: Input: Training examples  $\{\text{cand}(s_i), y_i^+\}_{i=1}^N$   $\triangleright y_i^+$  is the
   oracle tree for  $s_i$  among  $\text{cand}(s_i)$ 
2:  $\mathbf{w} \leftarrow \mathbf{0}$   $\triangleright$  initial weights
3: for  $t \leftarrow 1 \dots T$  do  $\triangleright T$  iterations
4:   for  $i \leftarrow 1 \dots N$  do
5:      $\hat{y} = \operatorname{argmax}_{y \in \text{cand}(s_i)} \mathbf{w} \cdot \mathbf{f}(y)$ 
6:     if  $\hat{y} \neq y_i^+$  then
7:        $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(y_i^+) - \mathbf{f}(\hat{y})$ 
8: return  $\mathbf{w}$ 

```

---

In  $n$ -best reranking, since all parses are explicitly enumerated, it is trivial to compute the oracle tree.<sup>2</sup> However, it remains widely open how to identify the *forest oracle*. We will present a dynamic programming algorithm for this problem in Sec. 4.1.

We also use a refinement called “averaged parameters” where the final weight vector is the average of weight vectors after each sentence in each iteration over the training data. This averaging effect has been shown to reduce overfitting and produce much more stable results (Collins, 2002).

### 3.2 Factorizing Local and Non-Local Features

A key difference between  $n$ -best and forest reranking is the handling of features. In  $n$ -best reranking, all features are treated equivalently by the decoder, which simply computes the value of each one on each candidate parse. However, for forest reranking, since the trees are not explicitly enumerated, many features can not be directly computed. So we first classify features into local and non-local, which the decoder will process in very different fashions.

We define a feature  $f$  to be **local** if and only if it can be factored among the local productions in a tree, and **non-local** if otherwise. For example, the **Rule** feature in Fig. 2(a) is local, while the **ParentRule** feature in Fig. 2(b) is non-local. It is worth noting that some features which seem complicated at the first sight are indeed local. For example, the **WordEdges** feature in Fig. 2(c), which classifies a node by its label, span length, and surrounding words, is still local since all these information are encoded either in the node itself or in the input sentence. In contrast, it would become non-local if we replace the surrounding words by surrounding POS

<sup>2</sup>In case multiple candidates get the same highest F-score, we choose the parse with the highest log probability from the baseline parser to be the oracle parse (Collins, 2000).



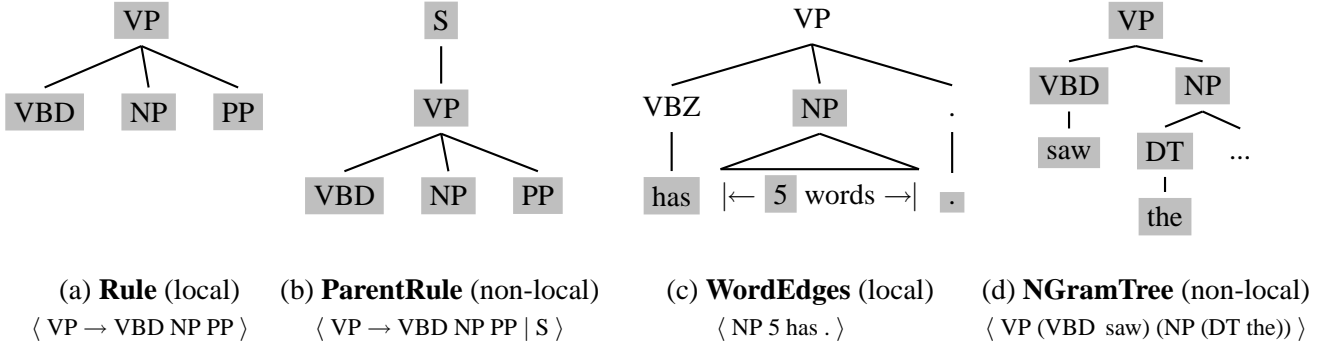


Figure 2: Illustration of some example features. Shaded nodes denote information included in the feature.

tags, which are generated dynamically.

More formally, we split the feature extractor  $\mathbf{f} = (f_1, \dots, f_d)$  into  $\mathbf{f} = (\mathbf{f}_L; \mathbf{f}_N)$  where  $\mathbf{f}_L$  and  $\mathbf{f}_N$  are the local and non-local features, respectively. For the former, we extend their domains from parses to hyperedges, where  $f(e)$  returns the value of a local feature  $f \in \mathbf{f}_L$  on hyperedge  $e$ , and its value on a parse  $y$  factors across the hyperedges (local productions),

$$\mathbf{f}_L(y) = \sum_{e \in y} \mathbf{f}_L(e) \quad (4)$$

and we can pre-compute  $\mathbf{f}_L(e)$  for each  $e$  in a forest.

Non-local features, however, can not be pre-computed, but we still prefer to compute them *early as possible*, which we call “on-the-fly” computation, so that our decoder can be sensitive to them at internal nodes. For instance, the **NGramTree** feature in Fig. 2 (d) returns the minimum tree fragmentation spanning a bigram, in this case “saw” and “the”, and should thus be computed at the *smallest common ancestor* of the two, which is the VP node in this example. Similarly, the **ParentRule** feature in Fig. 2 (b) can be computed when the S subtree is formed. In doing so, we essentially factor non-local features across *subtrees*, where for each subtree  $y'$  in a parse  $y$ , we define a **unit feature**  $\hat{f}(y')$  to be the part of  $f(y)$  that are computable within  $y'$ , but not computable in any (proper) subtree of  $y'$ . Then we have:

$$\mathbf{f}_N(y) = \sum_{y' \in y} \hat{\mathbf{f}}_N(y') \quad (5)$$

Intuitively, we compute the unit non-local features at each subtree from bottom-up. For example, for the binary-branching node  $A_{i,k}$  in Fig. 3, the

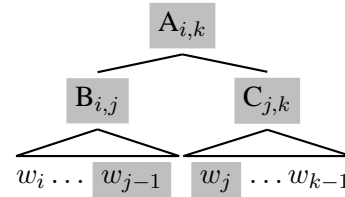


Figure 3: Example of the unit **NGramTree** feature at node  $A_{i,k}$ :  $\langle A (B \dots w_{j-1}) (C \dots w_j) \rangle$ .

unit **NGramTree** instance is for the pair  $\langle w_{j-1}, w_j \rangle$  on the boundary between the two subtrees, whose smallest common ancestor is the current node. Other unit **NGramTree** instances within this span have already been computed in the subtrees, except those for the boundary words of the whole node,  $w_i$  and  $w_{k-1}$ , which will be computed when this node is further combined with other nodes in the future.

### 3.3 Approximate Decoding via Cube Pruning

Before moving on to approximate decoding with non-local features, we first describe the algorithm for exact decoding when only local features are present, where many concepts and notations will be re-used later. We will use  $\mathbf{D}(v)$  to denote the top derivations of node  $v$ , where  $D_1(v)$  is its 1-best derivation. We also use the notation  $\langle e, \mathbf{j} \rangle$  to denote the derivation along hyperedge  $e$ , using the  $j_i$ th sub-derivation for tail  $u_i$ , so  $\langle e, \mathbf{1} \rangle$  is the best derivation along  $e$ . The exact decoding algorithm, shown in Pseudocode 2, is an instance of the bottom-up Viterbi algorithm, which traverses the hypergraph in a topological order, and at each node  $v$ , calculates its 1-best derivation using each incoming hyperedge  $e \in \text{IN}(v)$ . The cost of  $e$ ,  $c(e)$ , is the score of its

---

**Pseudocode 2** Exact Decoding with Local Features

---

```
1: function VITERBI( $\langle V, E \rangle$ )
2:   for  $v \in V$  in topological order do
3:     for  $e \in IN(v)$  do
4:        $c(e) \leftarrow \mathbf{w} \cdot \mathbf{f}_L(e) + \sum_{u_i \in tails(e)} c(D_1(u_i))$ 
5:       if  $c(e) > c(D_1(v))$  then  $\triangleright$  better derivation?
6:          $D_1(v) \leftarrow \langle e, \mathbf{1} \rangle$ 
7:          $c(D_1(v)) \leftarrow c(e)$ 
8:   return  $D_1(\text{TOP})$ 
```

---

---

**Pseudocode 3** Cube Pruning for Non-local Features

---

```
1: function CUBE( $\langle V, E \rangle$ )
2:   for  $v \in V$  in topological order do
3:     KBEST( $v$ )
4:   return  $D_1(\text{TOP})$ 
5: procedure KBEST( $v$ )
6:    $heap \leftarrow \emptyset$ ;  $buf \leftarrow \emptyset$ 
7:   for  $e \in IN(v)$  do
8:      $c(\langle e, \mathbf{1} \rangle) \leftarrow \text{EVAL}(e, \mathbf{1})$   $\triangleright$  extract unit features
9:      $\text{append}(\langle e, \mathbf{1} \rangle)$  to  $heap$ 
10:  HEAPIFY( $heap$ )  $\triangleright$  prioritized frontier
11:  while  $|heap| > 0$  and  $|buf| < k$  do
12:     $item \leftarrow \text{POP-MAX}(heap)$   $\triangleright$  extract next-best
13:     $\text{append}(item)$  to  $buf$ 
14:    PUSHSUCC( $item, heap$ )
15:  sort  $buf$  to  $\mathbf{D}(v)$ 
16: procedure PUSHSUCC( $\langle e, \mathbf{j} \rangle, heap$ )
17:    $e$  is  $v \rightarrow u_1 \dots u_{|e|}$ 
18:   for  $i$  in  $1 \dots |e|$  do
19:      $\mathbf{j}' \leftarrow \mathbf{j} + \mathbf{b}^i$   $\triangleright \mathbf{b}^i$  is 1 only on the  $i$ th dim.
20:     if  $|\mathbf{D}(u_i)| \geq j'_i$  then  $\triangleright$  enough sub-derivations?
21:        $c(\langle e, \mathbf{j}' \rangle) \leftarrow \text{EVAL}(e, \mathbf{j}')$   $\triangleright$  unit features
22:       PUSH( $\langle e, \mathbf{j}' \rangle, heap$ )
23: function EVAL( $e, \mathbf{j}$ )
24:    $e$  is  $v \rightarrow u_1 \dots u_{|e|}$ 
25:   return  $\mathbf{w} \cdot \mathbf{f}_L(e) + \mathbf{w} \cdot \mathbf{f}_N(\langle e, \mathbf{j} \rangle) + \sum_i c(D_{j_i}(u_i))$ 
```

---

(pre-computed) local features  $\mathbf{w} \cdot \mathbf{f}_L(e)$ . This algorithm has a time complexity of  $O(E)$ , and is almost identical to traditional chart parsing, except that the forest might be more than binary-branching.

For non-local features, we adapt cube pruning from forest rescoring (Chiang, 2007; Huang and Chiang, 2007), since the situation here is analogous to machine translation decoding with integrated language models: we can view the scores of unit non-local features as the language model cost, computed on-the-fly when combining sub-constituents.

Shown in Pseudocode 3, cube pruning works bottom-up on the forest, keeping a beam of at most  $k$  derivations at each node, and uses the  $k$ -best parsing Algorithm 2 of Huang and Chiang (2005) to speed up the computation. When combining the sub-

derivations along a hyperedge  $e$  to form a new subtree  $y' = \langle e, \mathbf{j} \rangle$ , we also compute its unit non-local feature values  $\mathbf{f}_N(\langle e, \mathbf{j} \rangle)$  (line 25). A priority queue (*heap* in Pseudocode 3) is used to hold the candidates for the next-best derivation, which is initialized to the set of best derivations along each hyperedge (lines 7 to 9). Then at each iteration, we pop the best derivation (lines 12), and push its successors back into the priority queue (line 14). Analogous to the language model cost in forest rescoring, the unit feature cost here is a non-monotonic score in the dynamic programming backbone, and the derivations may thus be extracted *out-of-order*. So a buffer *buf* is used to hold extracted derivations, which is sorted at the end (line 15) to form the list of top- $k$  derivations  $\mathbf{D}(v)$  of node  $v$ . The complexity of this algorithm is  $O(E + Vk \log k\mathcal{N})$  (Huang and Chiang, 2005), where  $O(\mathcal{N})$  is the time for on-the-fly feature extraction for each subtree, which becomes the bottleneck in practice.

## 4 Supporting Forest Algorithms

### 4.1 Forest Oracle

Recall that the Parseval F-score is the harmonic mean of labelled precision  $P$  and labelled recall  $R$ :

$$F(y, y^*) \triangleq \frac{2PR}{P+R} = \frac{2|y \cap y^*|}{|y| + |y^*|} \quad (6)$$

where  $|y|$  and  $|y^*|$  are the numbers of brackets in the test parse and gold parse, respectively, and  $|y \cap y^*|$  is the number of matched brackets. Since the harmonic mean is a non-linear combination, we can not optimize the F-scores on sub-forests independently with a greedy algorithm. In other words, the optimal F-score tree in a forest is *not* guaranteed to be composed of two optimal F-score subtrees.

We instead propose a dynamic programming algorithm which optimizes the number of matched brackets for a given number of test brackets. For example, our algorithm will ask questions like,

“when a test parse has 5 brackets, what is the maximum number of matched brackets?”

More formally, at each node  $v$ , we compute an *oracle function*  $ora[v] : \mathbb{N} \mapsto \mathbb{N}$ , which maps an integer  $t$  to  $ora[v](t)$ , the max. number of matched brackets

---

**Pseudocode 4** Forest Oracle Algorithm
 

---

```

1: function ORACLE( $\langle V, E \rangle, y^*$ )
2:   for  $v \in V$  in topological order do
3:     for  $e \in BS(v)$  do
4:        $e$  is  $v \rightarrow u_1 u_2 \dots u_{|e|}$ 
5:        $ora[v] \leftarrow ora[v] \oplus (\otimes_i ora[u_i])$ 
6:        $ora[v] \leftarrow ora[v] \uparrow (1, \mathbf{1}_{v \in y^*})$ 
7:   return  $F(y^+, y^*) = \max_t \frac{2 \cdot ora[\text{TOP}](t)}{t + |y^*|} \triangleright \text{oracle } F_1$ 

```

---

for all parses  $y_v$  of node  $v$  with exactly  $t$  brackets:

$$ora[v](t) \triangleq \max_{y_v: |y_v|=t} |y_v \cap y^*| \quad (7)$$

When node  $v$  is combined with another node  $u$  along a hyperedge  $e = \langle (v, u), w \rangle$ , we need to combine the two oracle functions  $ora[v]$  and  $ora[u]$  by distributing the test brackets of  $w$  between  $v$  and  $u$ , and optimize the number of matched brackets. To do this we define a *convolution operator*  $\otimes$  between two functions  $f$  and  $g$ :

$$(f \otimes g)(t) \triangleq \max_{t_1+t_2=t} f(t_1) + g(t_2) \quad (8)$$

For instance:

$$\begin{array}{c|c} t & f(t) \\ \hline 2 & 1 \\ 3 & 2 \end{array} \otimes \begin{array}{c|c} t & g(t) \\ \hline 4 & 4 \\ 5 & 4 \end{array} = \begin{array}{c|c} t & (f \otimes g)(t) \\ \hline 6 & 5 \\ 7 & 6 \\ 8 & 6 \end{array}$$

The oracle function for the head node  $w$  is then

$$ora[w](t) = (ora[v] \otimes ora[u])(t-1) + \mathbf{1}_{w \in y^*} \quad (9)$$

where  $\mathbf{1}$  is the indicator function, returning 1 if node  $w$  is found in the gold tree  $y^*$ , in which case we increment the number of matched brackets. We can also express Eq. 9 in a purely functional form

$$ora[w] = (ora[v] \otimes ora[u]) \uparrow (1, \mathbf{1}_{w \in y^*}) \quad (10)$$

where  $\uparrow$  is a *translation operator* which shifts a function along the axes:

$$(f \uparrow (a, b))(t) \triangleq f(t-a) + b \quad (11)$$

Above we discussed the case of one hyperedge. If there is another hyperedge  $e'$  deriving node  $w$ , we also need to combine the resulting oracle functions from both hyperedges, for which we define a *pointwise addition operator*  $\oplus$ :

$$(f \oplus g)(t) \triangleq \max\{f(t), g(t)\} \quad (12)$$

Shown in Pseudocode 4, we perform these computations in a bottom-up topological order, and finally at the root node TOP, we can compute the best global F-score by maximizing over different numbers of test brackets (line 7). The oracle tree  $y^+$  can be recursively restored by keeping backpointers for each  $ora[v](t)$ , which we omit in the pseudocode.

The time complexity of this algorithm for a sentence of  $l$  words is  $O(|E| \cdot l^{2(a-1)})$  where  $a$  is the arity of the forest. For a CKY forest, this amounts to  $O(l^3 \cdot l^2) = O(l^5)$ , but for general forests like those in our experiments the complexities are much higher. In practice it takes on average 0.05 seconds for forests pruned by  $p = 10$  (see Section 4.2), but we can pre-compute and store the oracle for each forest before training starts.

## 4.2 Forest Pruning

Our forest pruning algorithm (Jonathan Graehl, p.c.) is very similar to the method based on marginal probability (Charniak and Johnson, 2005), except that ours prunes hyperedges as well as nodes. Basically, we use an Inside-Outside algorithm to compute the Viterbi inside cost  $\beta(v)$  and the Viterbi outside cost  $\alpha(v)$  for each node  $v$ , and then compute the **merit**  $\alpha\beta(e)$  for each hyperedge:

$$\alpha\beta(e) = \alpha(\text{head}(e)) + \sum_{u_i \in \text{tails}(e)} \beta(u_i) \quad (13)$$

Intuitively, this merit is the cost of the best derivation that traverses  $e$ , and the difference  $\delta(e) = \alpha\beta(e) - \beta(\text{TOP})$  can be seen as the distance away from the globally best derivation. We prune away all hyperedges that have  $\delta(e) > p$  for a threshold  $p$ . Nodes with all incoming hyperedges pruned are also pruned. The key difference from (Charniak and Johnson, 2005) is that in this algorithm, a node can “partially” survive the beam, with a subset of its hyperedges pruned. In practice, this method prunes on average 15% more hyperedges than their method.

## 5 Experiments

We compare the performance of our forest reranker against  $n$ -best reranking on the Penn English Treebank (Marcus et al., 1993). The baseline parser is the Charniak parser, which we modified to output a

Local	instances	Non-Local	instances
<b>Rule</b>	10,851	<b>ParentRule</b>	18,019
<b>Word</b>	20,328	<b>WProj</b>	27,417
<b>WordEdges</b>	454,101	<b>Heads</b>	70,013
<b>CoLenPar</b>	22	<b>HeadTree</b>	67,836
<b>Bigram</b> <sup>◊</sup>	10,292	<b>Heavy</b>	1,401
<b>Trigram</b> <sup>◊</sup>	24,677	<b>NGramTree</b>	67,559
<b>HeadMod</b> <sup>◊</sup>	12,047	<b>RightBranch</b>	2
<b>DistMod</b> <sup>◊</sup>	16,017		
Total Feature Instances: 800,582			

Table 2: Features used in this work. Those with a <sup>◊</sup> are from (Collins, 2000), and others are from (Charniak and Johnson, 2005), with simplifications.

packed forest for each sentence.<sup>3</sup>

## 5.1 Data Preparation

We use the standard split of the Treebank: sections 02-21 as the training data (39832 sentences), section 22 as the development set (1700 sentences), and section 23 as the test set (2416 sentences). Following (Charniak and Johnson, 2005), the training set is split into 20 folds, each containing about 1992 sentences, and is parsed by the Charniak parser with a model trained on sentences from the remaining 19 folds. The development set and the test set are parsed with a model trained on all 39832 training sentences.

We implemented both  $n$ -best and forest reranking systems in Python and ran our experiments on a 64-bit Dual-Core Intel Xeon with 3.0GHz CPUs. Our feature set is summarized in Table 2, which closely follows Charniak and Johnson (2005), except that we excluded the non-local features **Edges**, **NGram**, and **CoPar**, and simplified **Rule** and **NGramTree** features, since they were too complicated to compute.<sup>4</sup> We also added four *unlexicalized* local features from Collins (2000) to cope with data-sparsity.

Following Charniak and Johnson (2005), we extracted the features from the 50-best parses on the training set (sec. 02-21), and used a cut-off of 5 to prune away low-count features. There are 0.8M features in our final set, considerably fewer than that of Charniak and Johnson which has about 1.3M fea-

<sup>3</sup>This is a relatively minor change to the Charniak parser, since it implements Algorithm 3 of Huang and Chiang (2005) for efficient enumeration of  $n$ -best parses, which requires storing the forest. The modified parser and related scripts for handling forests (e.g. oracles) will be available on my homepage.

<sup>4</sup>In fact, our **Rule** and **ParentRule** features are two special cases of the original **Rule** feature in (Charniak and Johnson, 2005). We also restricted **NGramTree** to be on bigrams only.

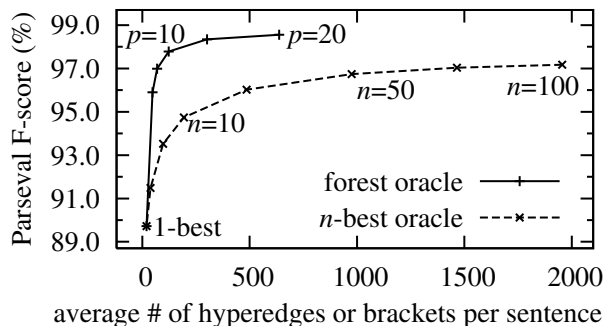


Figure 4: Forests (shown with various pruning thresholds) enjoy higher oracle scores and more compact sizes than  $n$ -best lists (on sec 23).

tures in the updated version.<sup>5</sup> However, our initial experiments show that, even with this much simpler feature set, our 50-best reranker performed equally well as theirs (both with an F-score of 91.4, see Tables 3 and 4). This result confirms that our feature set design is appropriate, and the averaged perceptron learner is a reasonable candidate for reranking.

The forests dumped from the Charniak parser are huge in size, so we use the forest pruning algorithm in Section 4.2 to prune them down to a reasonable size. In the following experiments we use a threshold of  $p = 10$ , which results in forests with an average number of 123.1 hyperedges per forest. Then for each forest, we annotate its forest oracle, and on each hyperedge, pre-compute its local features.<sup>6</sup> Shown in Figure 4, these forests have a forest oracle of 97.8, which is 1.1% higher than the 50-best oracle (96.7), and are 8 times smaller in size.

## 5.2 Results and Analysis

Table 3 compares the performance of forest reranking against standard  $n$ -best reranking. For both systems, we first use only the local features, and then all the features. We use the development set to determine the optimal number of iterations for averaged perceptron, and report the  $F_1$  score on the test set. With only local features, our forest reranker achieves an F-score of 91.25, and with the addition of non-

<sup>5</sup><http://www.cog.brown.edu/~mj/software.htm>. We follow this version as it corrects some bugs from their 2005 paper which leads to a 0.4% increase in performance (see Table 4).

<sup>6</sup>A subset of local features, e.g. **WordEdges**, is independent of which hyperedge the node takes in a derivation, and can thus be annotated on nodes rather than hyperedges. We call these features *node-local*, which also include part of **Word** features.

baseline: 1-best Charniak parser		89.72		
<i>n</i> -best reranking				
features	<i>n</i>	pre-comp.	training	F <sub>1</sub> %
local	50	1.7G / 16h	3 × 0.1h	91.28
all	50	2.4G / 19h	4 × 0.3h	91.43
all	100	5.3G / 44h	4 × 0.7h	91.49
forest reranking ( <i>p</i> = 10)				
features	<i>k</i>	pre-comp.	training	F <sub>1</sub> %
local	-	1.2G / 2.9h	3 × 0.8h	91.25
all	15		4 × 6.1h	<b>91.69</b>

Table 3: Forest reranking compared to *n*-best reranking on sec. 23. The **pre-comp.** column is for feature extraction, and **training** column shows the number of perceptron iterations that achieved best results on the dev set, and average time per iteration.

local features, the accuracy rises to 91.69 (with beam size  $k = 15$ ), which is a 0.26% absolute improvement over 50-best reranking.<sup>7</sup>

This improvement might look relatively small, but it is much harder to make a similar progress with *n*-best reranking. For example, even if we double the size of the *n*-best list to 100, the performance only goes up by 0.06% (Table 3). In fact, the 100-best oracle is only 0.5% higher than the 50-best one (see Fig. 4). In addition, the feature extraction step in 100-best reranking produces huge data files and takes 44 hours in total, though this part can be parallelized.<sup>8</sup> On two CPUs, 100-best reranking takes 25 hours, while our forest-reranker can also finish in 26 hours, with a much smaller disk space. Indeed, this demonstrates the severe redundancies as another disadvantage of *n*-best lists, where many subtrees are repeated across different parses, while the packed forest reduces space dramatically by sharing common sub-derivations (see Fig. 4).

To put our results in perspective, we also compare them with other best-performing systems in Table 4. Our final result (91.7) is better than any previously reported system trained on the Treebank, although

<sup>7</sup>It is surprising that 50-best reranking with local features achieves an even higher F-score of 91.28, and we suspect this is due to the aggressive updates and instability of the perceptron, as we do observe the learning curves to be non-monotonic. We leave the use of more stable learning algorithms to future work.

<sup>8</sup>The *n*-best feature extraction already uses *relative counts* (Johnson, 2006), which reduced file sizes by at least a factor 4.

type	system	F <sub>1</sub> %
D	Collins (2000)	89.7
	Henderson (2004)	90.1
	Charniak and Johnson (2005)	91.0
	<i>updated</i> (Johnson, 2006)	91.4
	<b>this work</b>	<b>91.7</b>
G	Bod (2003)	90.7
	Petrov and Klein (2007)	90.1
S	McClosky et al. (2006)	<b>92.1</b>

Table 4: Comparison of our final results with other best-performing systems on the whole Section 23. Types D, G, and S denote discriminative, generative, and semi-supervised approaches, respectively.

McClosky et al. (2006) achieved an even higher accuracy (92.1) by leveraging on much larger unlabelled data. Moreover, their technique is orthogonal to ours, and we suspect that replacing their *n*-best reranker by our forest reranker might get an even better performance. Plus, except for *n*-best reranking, most discriminative methods require repeated parsing of the training set, which is generally impractical (Petrov and Klein, 2008). Therefore, previous work often resorts to extremely short sentences ( $\leq 15$  words) or only looked at local features (Taskar et al., 2004; Henderson, 2004; Turian and Melamed, 2007). In comparison, thanks to the efficient decoding, our work not only scaled to the whole Treebank, but also successfully incorporated non-local features, which showed an absolute improvement of 0.44% over that of local features alone.

## 6 Conclusion

We have presented a framework for reranking on packed forests which compactly encodes many more candidates than *n*-best lists. With efficient approximate decoding, perceptron training on the whole Treebank becomes practical, which can be done in about a day even with a Python implementation. Our final result outperforms both 50-best and 100-best reranking baselines, and is better than any previously reported systems trained on the Treebank. We also devised a dynamic programming algorithm for forest oracles, an interesting problem by itself. We believe this general framework could also be applied to other problems involving forests or lattices, such as sequence labeling and machine translation.

## References

- Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of ACL '89*, pages 143–151.
- Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of EACL*.
- Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine-grained  $n$ -best parsing and discriminative reranking. In *Proceedings of the 43rd ACL*.
- Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*.
- David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–208.
- Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML*, pages 175–182.
- Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.
- James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of ACL*.
- Liang Huang and David Chiang. 2005. Better  $k$ -best Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-2005)*.
- Liang Huang and David Chiang. 2007. Forest rescoring: Fast decoding with integrated language models. In *Proceedings of ACL*.
- Mark Johnson. 2006. Features of statistical parsers. Talk given at the *Joint Microsoft Research and Univ. of Washington Computational Linguistics Colloquium*. <http://www.cog.brown.edu/~mj/papers/ms-uw06talk.pdf>.
- Dan Klein and Christopher D. Manning. 2001. Parsing and Hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001), 17-19 October 2001, Beijing, China*.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.
- David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the HLT-NAACL*, New York City, USA, June.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd ACL*.
- Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*.
- Slav Petrov and Dan Klein. 2008. Discriminative log-linear grammars with latent variables. In *Proceedings of NIPS 20*.
- Libin Shen, Anoop Sarkar, and Franz Josef Och. 2005. Discriminative reranking for machine translation. In *Proceedings of HLT-NAACL*.
- Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Chris Manning. 2004. Max-margin parsing. In *Proceedings of EMNLP*.
- Joseph Turian and I. Dan Melamed. 2007. Scalable discriminative learning for natural language parsing and translation. In *Proceedings of NIPS 19*.

# Simple Semi-supervised Dependency Parsing

Terry Koo, Xavier Carreras, and Michael Collins

MIT CSAIL, Cambridge, MA 02139, USA

{maestro, carreras, mcollins}@csail.mit.edu

## Abstract

We present a simple and effective semi-supervised method for training dependency parsers. We focus on the problem of lexical representation, introducing features that incorporate word clusters derived from a large unannotated corpus. We demonstrate the effectiveness of the approach in a series of dependency parsing experiments on the Penn Treebank and Prague Dependency Treebank, and we show that the cluster-based features yield substantial gains in performance across a wide range of conditions. For example, in the case of English unlabeled second-order parsing, we improve from a baseline accuracy of 92.02% to 93.16%, and in the case of Czech unlabeled second-order parsing, we improve from a baseline accuracy of 86.13% to 87.13%. In addition, we demonstrate that our method also improves performance when small amounts of training data are available, and can roughly halve the amount of supervised data required to reach a desired level of performance.

## 1 Introduction

In natural language parsing, lexical information is seen as crucial to resolving ambiguous relationships, yet lexicalized statistics are sparse and difficult to estimate directly. It is therefore attractive to consider intermediate entities which exist at a coarser level than the words themselves, yet capture the information necessary to resolve the relevant ambiguities.

In this paper, we introduce lexical intermediaries via a simple two-stage semi-supervised approach. First, we use a large unannotated corpus to define word clusters, and then we use that clustering to construct a new cluster-based feature mapping for a discriminative learner. We are thus relying on the ability of discriminative learning methods to identify

and exploit informative features while remaining agnostic as to the origin of such features. To demonstrate the effectiveness of our approach, we conduct experiments in dependency parsing, which has been the focus of much recent research—e.g., see work in the CoNLL shared tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007).

The idea of combining word clusters with discriminative learning has been previously explored by Miller et al. (2004), in the context of named-entity recognition, and their work directly inspired our research. However, our target task of dependency parsing involves more complex structured relationships than named-entity tagging; moreover, it is not at all clear that word clusters should have any relevance to syntactic structure. Nevertheless, our experiments demonstrate that word clusters can be quite effective in dependency parsing applications.

In general, semi-supervised learning can be motivated by two concerns: first, given a fixed amount of supervised data, we might wish to leverage additional unlabeled data to facilitate the utilization of the supervised corpus, increasing the performance of the model in absolute terms. Second, given a fixed target performance level, we might wish to use unlabeled data to reduce the amount of annotated data necessary to reach this target.

We show that our semi-supervised approach yields improvements for fixed datasets by performing parsing experiments on the Penn Treebank (Marcus et al., 1993) and Prague Dependency Treebank (Hajič, 1998; Hajič et al., 2001) (see Sections 4.1 and 4.3). By conducting experiments on datasets of varying sizes, we demonstrate that for fixed levels of performance, the cluster-based approach can reduce the need for supervised data by roughly half, which is a substantial savings in data-annotation costs (see Sections 4.2 and 4.4).

The remainder of this paper is divided as follows:

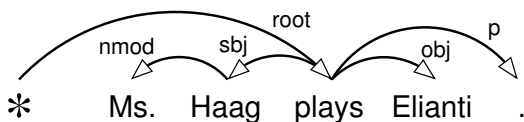


Figure 1: An example of a labeled dependency tree. The tree contains a special token “\*” which is always the root of the tree. Each arc is directed from head to modifier and has a label describing the function of the attachment.

Section 2 gives background on dependency parsing and clustering, Section 3 describes the cluster-based features, Section 4 presents our experimental results, Section 5 discusses related work, and Section 6 concludes with ideas for future research.

## 2 Background

### 2.1 Dependency parsing

Recent work (Buchholz and Marsi, 2006; Nivre et al., 2007) has focused on dependency parsing. Dependency syntax represents syntactic information as a network of head-modifier dependency arcs, typically restricted to be a directed tree (see Figure 1 for an example). Dependency parsing depends critically on predicting head-modifier relationships, which can be difficult due to the statistical sparsity of these word-to-word interactions. Bilexical dependencies are thus ideal candidates for the application of coarse word proxies such as word clusters.

In this paper, we take a part-factored structured classification approach to dependency parsing. For a given sentence  $\mathbf{x}$ , let  $\mathcal{Y}(\mathbf{x})$  denote the set of possible dependency structures spanning  $\mathbf{x}$ , where each  $y \in \mathcal{Y}(\mathbf{x})$  decomposes into a set of “parts”  $r \in y$ . In the simplest case, these parts are the dependency arcs themselves, yielding a first-order or “edge-factored” dependency parsing model. In higher-order parsing models, the parts can consist of interactions between more than two words. For example, the parser of McDonald and Pereira (2006) defines parts for sibling interactions, such as the trio “plays”, “Elianti”, and “.” in Figure 1. The Carreras (2007) parser has parts for both sibling interactions and grandparent interactions, such as the trio “\*”, “plays”, and “Haag” in Figure 1. These kinds of higher-order factorizations allow dependency parsers to obtain a limited form of context-sensitivity.

Given a factorization of dependency structures into parts, we restate dependency parsing as the fol-

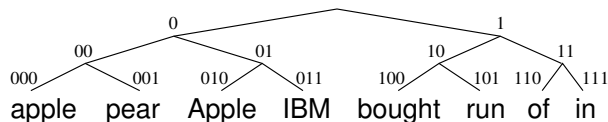


Figure 2: An example of a Brown word-cluster hierarchy. Each node in the tree is labeled with a bit-string indicating the path from the root node to that node, where 0 indicates a left branch and 1 indicates a right branch.

lowing maximization:

$$\text{PARSE}(\mathbf{x}; \mathbf{w}) = \operatorname{argmax}_{y \in \mathcal{Y}(\mathbf{x})} \sum_{r \in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, r)$$

Above, we have assumed that each part is scored by a linear model with parameters  $\mathbf{w}$  and feature-mapping  $\mathbf{f}(\cdot)$ . For many different part factorizations and structure domains  $\mathcal{Y}(\cdot)$ , it is possible to solve the above maximization efficiently, and several recent efforts have concentrated on designing new maximization algorithms with increased context-sensitivity (Eisner, 2000; McDonald et al., 2005b; McDonald and Pereira, 2006; Carreras, 2007).

### 2.2 Brown clustering algorithm

In order to provide word clusters for our experiments, we used the Brown clustering algorithm (Brown et al., 1992). We chose to work with the Brown algorithm due to its simplicity and prior success in other NLP applications (Miller et al., 2004; Liang, 2005). However, we expect that our approach can function with other clustering algorithms (as in, e.g., Li and McCallum (2005)). We briefly describe the Brown algorithm below.

The input to the algorithm is a vocabulary of words to be clustered and a corpus of text containing these words. Initially, each word in the vocabulary is considered to be in its own distinct cluster. The algorithm then repeatedly merges the pair of clusters which causes the smallest decrease in the likelihood of the text corpus, according to a class-based bigram language model defined on the word clusters. By tracing the pairwise merge operations, one obtains a hierarchical clustering of the words, which can be represented as a binary tree as in Figure 2.

Within this tree, each word is uniquely identified by its path from the root, and this path can be compactly represented with a bit string, as in Figure 2. In order to obtain a clustering of the words, we select all nodes at a certain depth from the root of the



hierarchy. For example, in Figure 2 we might select the four nodes at depth 2 from the root, yielding the clusters {apple,pear}, {Apple,IBM}, {bought,run}, and {of,in}. Note that the same clustering can be obtained by truncating each word’s bit-string to a 2-bit prefix. By using prefixes of various lengths, we can produce clusterings of different granularities (Miller et al., 2004).

For all of the experiments in this paper, we used the Liang (2005) implementation of the Brown algorithm to obtain the necessary word clusters.

### 3 Feature design

Key to the success of our approach is the use of features which allow word-cluster-based information to assist the parser. The feature sets we used are similar to other feature sets in the literature (McDonald et al., 2005a; Carreras, 2007), so we will not attempt to give a exhaustive description of the features in this section. Rather, we describe our features at a high level and concentrate on our methodology and motivations. In our experiments, we employed two different feature sets: a baseline feature set which draws upon “normal” information sources such as word forms and parts of speech, and a cluster-based feature set that also uses information derived from the Brown cluster hierarchy.

#### 3.1 Baseline features

Our first-order baseline feature set is similar to the feature set of McDonald et al. (2005a), and consists of indicator functions for combinations of words and parts of speech for the head and modifier of each dependency, as well as certain contextual tokens.<sup>1</sup> Our second-order baseline features are the same as those of Carreras (2007) and include indicators for triples of part of speech tags for sibling interactions and grandparent interactions, as well as additional bigram features based on pairs of words involved these higher-order interactions. Examples of baseline features are provided in Table 1.

<sup>1</sup>We augment the McDonald et al. (2005a) feature set with backed-off versions of the “Surrounding Word POS Features” that include only one neighboring POS tag. We also add binned distance features which indicate whether the number of tokens between the head and modifier of a dependency is greater than 2, 5, 10, 20, 30, or 40 tokens.

Baseline	Cluster-based
ht, mt	hc4, mc4
hw, mw	hc6, mc6
hw, ht, mt	hc*, mc*
hw, ht, mw	hc4, mt
ht, mw, mt	ht, mc4
hw, mw, mt	hc6, mt
hw, ht, mw, mt	ht, mc6
...	hc4, mw
	hw, mc4
	...
ht, mt, st	hc4, mc4, sc4
ht, mt, gt	hc6, mc6, sc6
...	ht, mc4, sc4
	hc4, mc4, gc4
	...

Table 1: Examples of baseline and cluster-based feature templates. Each entry represents a class of indicators for tuples of information. For example, “ht, mt” represents a class of indicator features with one feature for each possible combination of head POS-tag and modifier POS-tag. Abbreviations: ht = head POS, hw = head word, hc4 = 4-bit prefix of head, hc6 = 6-bit prefix of head, hc\* = full bit string of head; mt, mw, mc4, mc6, mc\* = likewise for modifier; st, gt, sc4, gc4, ... = likewise for sibling and grandchild.

#### 3.2 Cluster-based features

The first- and second-order cluster-based feature sets are supersets of the baseline feature sets: they include all of the baseline feature templates, and add an additional layer of features that incorporate word clusters. Following Miller et al. (2004), we use prefixes of the Brown cluster hierarchy to produce clusterings of varying granularity. We found that it was nontrivial to select the proper prefix lengths for the dependency parsing task; in particular, the prefix lengths used in the Miller et al. (2004) work (between 12 and 20 bits) performed poorly in dependency parsing.<sup>2</sup> After experimenting with many different feature configurations, we eventually settled on a simple but effective methodology.

First, we found that it was helpful to employ two different types of word clusters:

1. Short bit-string prefixes (e.g., 4–6 bits), which we used as replacements for parts of speech.

<sup>2</sup>One possible explanation is that the kinds of distinctions required in a named-entity recognition task (e.g., “Alice” versus “Intel”) are much finer-grained than the kinds of distinctions relevant to syntax (e.g., “apple” versus “eat”).

2. Full bit strings,<sup>3</sup> which we used as substitutes for word forms.

Using these two types of clusters, we generated new features by mimicking the template structure of the original baseline features. For example, the baseline feature set includes indicators for word-to-word and tag-to-tag interactions between the head and modifier of a dependency. In the cluster-based feature set, we correspondingly introduce new indicators for interactions between pairs of short bit-string prefixes and pairs of full bit strings. Some examples of cluster-based features are given in Table 1.

Second, we found it useful to concentrate on “hybrid” features involving, e.g., one bit-string and one part of speech. In our initial attempts, we focused on features that used cluster information exclusively. While these cluster-only features provided some benefit, we found that adding hybrid features resulted in even greater improvements. One possible explanation is that the clusterings generated by the Brown algorithm can be noisy or only weakly relevant to syntax; thus, the clusters are best exploited when “anchored” to words or parts of speech.

Finally, we found it useful to impose a form of vocabulary restriction on the cluster-based features. Specifically, for any feature that is predicated on a word form, we eliminate this feature if the word in question is *not* one of the top- $N$  most frequent words in the corpus. When  $N$  is between roughly 100 and 1,000, there is little effect on the performance of the cluster-based feature sets.<sup>4</sup> In addition, the vocabulary restriction reduces the size of the feature sets to manageable proportions.

## 4 Experiments

In order to evaluate the effectiveness of the cluster-based feature sets, we conducted dependency parsing experiments in English and Czech. We test the features in a wide range of parsing configurations, including first-order and second-order parsers, and labeled and unlabeled parsers.<sup>5</sup>

<sup>3</sup>As in Brown et al. (1992), we limit the clustering algorithm so that it recovers at most 1,000 distinct bit-strings; thus full bit strings are not equivalent to word forms.

<sup>4</sup>We used  $N = 800$  for all experiments in this paper.

<sup>5</sup>In an “unlabeled” parser, we simply ignore dependency label information, which is a common simplification.

The English experiments were performed on the Penn Treebank (Marcus et al., 1993), using a standard set of head-selection rules (Yamada and Matsumoto, 2003) to convert the phrase structure syntax of the Treebank to a dependency tree representation.<sup>6</sup> We split the Treebank into a training set (Sections 2–21), a development set (Section 22), and several test sets (Sections 0,<sup>7</sup> 1, 23, and 24). The data partition and head rules were chosen to match previous work (Yamada and Matsumoto, 2003; McDonald et al., 2005a; McDonald and Pereira, 2006). The part of speech tags for the development and test data were automatically assigned by MXPOST (Ratnaparkhi, 1996), where the tagger was trained on the entire training corpus; to generate part of speech tags for the training data, we used 10-way jackknifing.<sup>8</sup> English word clusters were derived from the BLLIP corpus (Charniak et al., 2000), which contains roughly 43 million words of Wall Street Journal text.<sup>9</sup>

The Czech experiments were performed on the Prague Dependency Treebank 1.0 (Hajič, 1998; Hajič et al., 2001), which is directly annotated with dependency structures. To facilitate comparisons with previous work (McDonald et al., 2005b; McDonald and Pereira, 2006), we used the training/development/test partition defined in the corpus and we also used the automatically-assigned part of speech tags provided in the corpus.<sup>10</sup> Czech word clusters were derived from the raw text section of the PDT 1.0, which contains about 39 million words of newswire text.<sup>11</sup>

We trained the parsers using the averaged perceptron (Freund and Schapire, 1999; Collins, 2002), which represents a balance between strong performance and fast training times. To select the number

<sup>6</sup>We used Joakim Nivre’s “Penn2Malt” conversion tool (<http://w3.msi.vxu.se/nivre/research/Penn2Malt.html>). Dependency labels were obtained via the “Malt” hard-coded setting.

<sup>7</sup>For computational reasons, we removed a single 249-word sentence from Section 0.

<sup>8</sup>That is, we tagged each fold with the tagger trained on the other 9 folds.

<sup>9</sup>We ensured that the sentences of the Penn Treebank were excluded from the text used for the clustering.

<sup>10</sup>Following Collins et al. (1999), we used a coarsened version of the Czech part of speech tags; this choice also matches the conditions of previous work (McDonald et al., 2005b; McDonald and Pereira, 2006).

<sup>11</sup>This text was disjoint from the training and test corpora.

Sec	dep1	dep1c	MD1	dep2	dep2c	MD2	dep1-L	dep1c-L	dep2-L	dep2c-L
00	90.48	91.57 (+1.09)	—	91.76	92.77 (+1.01)	—	90.29	91.03 (+0.74)	91.33	92.09 (+0.76)
01	91.31	92.43 (+1.12)	—	92.46	93.34 (+0.88)	—	90.84	91.73 (+0.89)	91.94	92.65 (+0.71)
23	90.84	92.23 (+1.39)	90.9	92.02	93.16 (+1.14)	91.5	90.32	91.24 (+0.92)	91.38	92.14 (+0.76)
24	89.67	91.30 (+1.63)	—	90.92	91.85 (+0.93)	—	89.55	90.06 (+0.51)	90.42	91.18 (+0.76)

Table 2: Parent-prediction accuracies on Sections 0, 1, 23, and 24. Abbreviations: dep1/dep1c = first-order parser with baseline/cluster-based features; dep2/dep2c = second-order parser with baseline/cluster-based features; MD1 = McDonald et al. (2005a); MD2 = McDonald and Pereira (2006); suffix -L = labeled parser. Unlabeled parsers are scored using unlabeled parent predictions, and labeled parsers are scored using labeled parent predictions. Improvements of cluster-based features over baseline features are shown in parentheses.

of iterations of perceptron training, we performed up to 30 iterations and chose the iteration which optimized accuracy on the development set. Our feature mappings are quite high-dimensional, so we eliminated all features which occur only once in the training data. The resulting models still had very high dimensionality, ranging from tens of millions to as many as a billion features.<sup>12</sup>

All results presented in this section are given in terms of parent-prediction accuracy, which measures the percentage of tokens that are attached to the correct head token. For labeled dependency structures, both the head token and dependency label must be correctly predicted. In addition, in English parsing we ignore the parent-predictions of punctuation tokens,<sup>13</sup> and in Czech parsing we retain the punctuation tokens; this matches previous work (Yamada and Matsumoto, 2003; McDonald et al., 2005a; McDonald and Pereira, 2006).

#### 4.1 English main results

In our English experiments, we tested eight different parsing configurations, representing all possible choices between baseline or cluster-based feature sets, first-order (Eisner, 2000) or second-order (Carreras, 2007) factorizations, and labeled or unlabeled parsing.

Table 2 compiles our final test results and also includes two results from previous work by McDonald et al. (2005a) and McDonald and Pereira (2006), for the purposes of comparison. We note a few small differences between our parsers and the

parsers evaluated in this previous work. First, the MD1 and MD2 parsers were trained via the MIRA algorithm (Crammer and Singer, 2003; Crammer et al., 2004), while we use the averaged perceptron. In addition, the MD2 model uses only sibling interactions, whereas the dep2/dep2c parsers include both sibling and grandparent interactions.

There are some clear trends in the results of Table 2. First, performance increases with the order of the parser: edge-factored models (dep1 and MD1) have the lowest performance, adding sibling relationships (MD2) increases performance, and adding grandparent relationships (dep2) yields even better accuracies. Similar observations regarding the effect of model order have also been made by Carreras (2007).

Second, note that the parsers using cluster-based feature sets consistently outperform the models using the baseline features, regardless of model order or label usage. Some of these improvements can be quite large; for example, a first-order model using cluster-based features generally performs as well as a second-order model using baseline features. Moreover, the benefits of cluster-based feature sets combine additively with the gains of increasing model order. For example, consider the unlabeled parsers in Table 2: on Section 23, increasing the model order from dep1 to dep2 results in a relative reduction in error of roughly 13%, while introducing cluster-based features from dep2 to dep2c yields an additional relative error reduction of roughly 14%. As a final note, all 16 comparisons between cluster-based features and baseline features shown in Table 2 are statistically significant.<sup>14</sup>

<sup>12</sup>Due to the sparsity of the perceptron updates, however, only a small fraction of the possible features were active in our trained models.

<sup>13</sup>A punctuation token is any token whose gold-standard part of speech tag is one of { ` ` ' ' : , . }.

<sup>14</sup>We used the sign test at the sentence level. The comparison between dep1-L and dep1c-L is significant at  $p < 0.05$ , and all other comparisons are significant at  $p < 0.0005$ .

Tagger always trained on full Treebank							Tagger trained on reduced dataset						
Size	dep1	dep1c	$\Delta$	dep2	dep2c	$\Delta$	Size	dep1	dep1c	$\Delta$	dep2	dep2c	$\Delta$
1k	84.54	85.90	1.36	86.29	87.47	1.18	1k	80.49	84.06	3.57	81.95	85.33	3.38
2k	86.20	87.65	1.45	87.67	88.88	1.21	2k	83.47	86.04	2.57	85.02	87.54	2.52
4k	87.79	89.15	1.36	89.22	90.46	1.24	4k	86.53	88.39	1.86	87.88	89.67	1.79
8k	88.92	90.22	1.30	90.62	91.55	0.93	8k	88.25	89.94	1.69	89.71	91.37	1.66
16k	90.00	91.27	1.27	91.27	92.39	1.12	16k	89.66	91.03	1.37	91.14	92.22	1.08
32k	90.74	92.18	1.44	92.05	93.36	1.31	32k	90.78	92.12	1.34	92.09	93.21	1.12
All	90.89	92.33	1.44	92.42	93.30	0.88	All	90.89	92.33	1.44	92.42	93.30	0.88

Table 3: Parent-prediction accuracies of unlabeled English parsers on Section 22. Abbreviations: Size = #sentences in training corpus;  $\Delta$  = difference between cluster-based and baseline features; other abbreviations are as in Table 2.

## 4.2 English learning curves

We performed additional experiments to evaluate the effect of the cluster-based features as the amount of training data is varied. Note that the dependency parsers we use require the input to be tagged with parts of speech; thus the quality of the part-of-speech tagger can have a strong effect on the performance of the parser. In these experiments, we consider two possible scenarios:

1. The tagger has a large training corpus, while the parser has a smaller training corpus. This scenario can arise when tagged data is cheaper to obtain than syntactically-annotated data.
2. The same amount of labeled data is available for training both tagger and parser.

Table 3 displays the accuracy of first- and second-order models when trained on smaller portions of the Treebank, in both scenarios described above. Note that the cluster-based features obtain consistent gains regardless of the size of the training set. When the tagger is trained on the reduced-size datasets, the gains of cluster-based features are more pronounced, but substantial improvements are obtained even when the tagger is accurate.

It is interesting to consider the amount by which cluster-based features reduce the need for supervised data, given a desired level of accuracy. Based on Table 3, we can extrapolate that cluster-based features reduce the need for supervised data by roughly a factor of 2. For example, the performance of the dep1c and dep2c models trained on 1k sentences is roughly the same as the performance of the dep1 and dep2 models, respectively, trained on 2k sentences. This approximate data-halving effect can be

observed throughout the results in Table 3.

When combining the effects of model order and cluster-based features, the reductions in the amount of supervised data required are even larger. For example, in scenario 1 the dep2c model trained on 1k sentences is close in performance to the dep1 model trained on 4k sentences, and the dep2c model trained on 4k sentences is close to the dep1 model trained on the entire training set (roughly 40k sentences).

## 4.3 Czech main results

In our Czech experiments, we considered only unlabeled parsing,<sup>15</sup> leaving four different parsing configurations: baseline or cluster-based features and first-order or second-order parsing. Note that our feature sets were originally tuned for English parsing, and except for the use of Czech clusters, we made no attempt to retune our features for Czech.

Czech dependency structures may contain non-projective edges, so we employ a maximum directed spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967; McDonald et al., 2005b) as our first-order parser for Czech. For the second-order parsing experiments, we used the Carreras (2007) parser. Since this parser only considers projective dependency structures, we “projectivized” the PDT 1.0 training set by finding, for each sentence, the projective tree which retains the most correct dependencies; our second-order parsers were then trained with respect to these projective trees. The development and test sets were *not* projectivized, so our second-order parser is guaranteed to make errors in test sentences containing non-projective dependencies. To overcome this, McDonald and Pereira (2006) use a

<sup>15</sup>We leave labeled parsing experiments to future work.

dep1	dep1c	dep2	dep2c
84.49	86.07 (+1.58)	86.13	87.13 (+1.00)

Table 4: Parent-prediction accuracies of unlabeled Czech parsers on the PDT 1.0 test set, for baseline features and cluster-based features. Abbreviations are as in Table 2.

Parser	Accuracy
Nivre and Nilsson (2005)	80.1
McDonald et al. (2005b)	84.4
Hall and Novák (2005)	85.1
McDonald and Pereira (2006)	85.2
dep1c	86.07
dep2c	87.13

Table 5: Unlabeled parent-prediction accuracies of Czech parsers on the PDT 1.0 test set, for our models and for previous work.

Size	dep1	dep1c	$\Delta$	dep2	dep2c	$\Delta$
1k	72.79	73.66	0.87	74.35	74.63	0.28
2k	74.92	76.23	1.31	76.63	77.60	0.97
4k	76.87	78.14	1.27	78.34	79.34	1.00
8k	78.17	79.83	1.66	79.82	80.98	1.16
16k	80.60	82.44	1.84	82.53	83.69	1.16
32k	82.85	84.65	1.80	84.66	85.81	1.15
64k	84.20	85.98	1.78	86.01	87.11	1.10
All	84.36	86.09	1.73	86.09	87.26	1.17

Table 6: Parent-prediction accuracies of unlabeled Czech parsers on the PDT 1.0 development set. Abbreviations are as in Table 3.

two-stage approximate decoding process in which the output of their second-order parser is “deprojectivized” via greedy search. For simplicity, we did not implement a deprojectivization stage on top of our second-order parser, but we conjecture that such techniques may yield some additional performance gains; we leave this to future work.

Table 4 gives accuracy results on the PDT 1.0 test set for our unlabeled parsers. As in the English experiments, there are clear trends in the results: parsers using cluster-based features outperform parsers using baseline features, and second-order parsers outperform first-order parsers. Both of the comparisons between cluster-based and baseline features in Table 4 are statistically significant.<sup>16</sup> Table 5 compares accuracy results on the PDT 1.0 test set for our parsers and several other recent papers.

<sup>16</sup>We used the sign test at the sentence level; both comparisons are significant at  $p < 0.0005$ .

$N$	dep1	dep1c	dep2	dep2c
100	89.19	92.25	90.61	93.14
200	90.03	92.26	91.35	93.18
400	90.31	92.32	91.72	93.20
800	90.62	92.33	91.89	93.30
1600	90.87	—	92.20	—
All	90.89	—	92.42	—

Table 7: Parent-prediction accuracies of unlabeled English parsers on Section 22. Abbreviations:  $N$  = threshold value; other abbreviations are as in Table 2. We did not train cluster-based parsers using threshold values larger than 800 due to computational limitations.

dep1-P	dep1c-P	dep1	dep2-P	dep2c-P	dep2
77.19	90.69	90.89	86.73	91.84	92.42

Table 8: Parent-prediction accuracies of unlabeled English parsers on Section 22. Abbreviations: suffix -P = model without POS; other abbreviations are as in Table 2.

#### 4.4 Czech learning curves

As in our English experiments, we performed additional experiments on reduced sections of the PDT; the results are shown in Table 6. For simplicity, we did not retrain a tagger for each reduced dataset, so we always use the (automatically-assigned) part of speech tags provided in the corpus. Note that the cluster-based features obtain improvements at all training set sizes, with data-reduction factors similar to those observed in English. For example, the dep1c model trained on 4k sentences is roughly as good as the dep1 model trained on 8k sentences.

#### 4.5 Additional results

Here, we present two additional results which further explore the behavior of the cluster-based feature sets. In Table 7, we show the development-set performance of second-order parsers as the threshold for lexical feature elimination (see Section 3.2) is varied. Note that the performance of cluster-based features is fairly insensitive to the threshold value, whereas the performance of baseline features clearly degrades as the vocabulary size is reduced.

In Table 8, we show the development-set performance of the first- and second-order parsers when features containing part-of-speech-based information are eliminated. Note that the performance obtained by using clusters *without* parts of speech is close to the performance of the baseline features.

## 5 Related Work

As mentioned earlier, our approach was inspired by the success of Miller et al. (2004), who demonstrated the effectiveness of using word clusters as features in a discriminative learning approach. Our research, however, applies this technique to dependency parsing rather than named-entity recognition.

In this paper, we have focused on developing new representations for lexical information. Previous research in this area includes several models which incorporate hidden variables (Matsuzaki et al., 2005; Koo and Collins, 2005; Petrov et al., 2006; Titov and Henderson, 2007). These approaches have the advantage that the model is able to learn different usages for the hidden variables, depending on the target problem at hand. Crucially, however, these methods do not exploit unlabeled data when learning their representations.

Wang et al. (2005) used distributional similarity scores to smooth a generative probability model for dependency parsing and obtained improvements in a Chinese parsing task. Our approach is similar to theirs in that the Brown algorithm produces clusters based on distributional similarity, and the cluster-based features can be viewed as being a kind of “backed-off” version of the baseline features. However, our work is focused on discriminative learning as opposed to generative models.

Semi-supervised phrase structure parsing has been previously explored by McClosky et al. (2006), who applied a reranked parser to a large unsupervised corpus in order to obtain additional training data for the parser; this self-training approach was shown to be quite effective in practice. However, their approach depends on the usage of a high-quality parse reranker, whereas the method described here simply augments the features of an existing parser. Note that our two approaches are compatible in that we could also design a reranker and apply self-training techniques on top of the cluster-based features.

## 6 Conclusions

In this paper, we have presented a simple but effective semi-supervised learning approach and demonstrated that it achieves substantial improvement over a competitive baseline in two broad-coverage depen-

ency parsing tasks. Despite this success, there are several ways in which our approach might be improved.

To begin, recall that the Brown clustering algorithm is based on a bigram language model. Intuitively, there is a “mismatch” between the kind of lexical information that is captured by the Brown clusters and the kind of lexical information that is modeled in dependency parsing. A natural avenue for further research would be the development of clustering algorithms that reflect the syntactic behavior of words; e.g., an algorithm that attempts to maximize the likelihood of a treebank, according to a probabilistic dependency model. Alternately, one could design clustering algorithms that cluster entire head-modifier arcs rather than individual words.

Another idea would be to integrate the clustering algorithm into the training algorithm in a limited fashion. For example, after training an initial parser, one could parse a large amount of unlabeled text and use those parses to improve the quality of the clusters. These improved clusters can then be used to retrain an improved parser, resulting in an overall algorithm similar to that of McClosky et al. (2006).

Setting aside the development of new clustering algorithms, a final area for future work is the extension of our method to new domains, such as conversational text or other languages, and new NLP problems, such as machine translation.

## Acknowledgments

The authors thank the anonymous reviewers for their insightful comments. Many thanks also to Percy Liang for providing his implementation of the Brown algorithm, and Ryan McDonald for his assistance with the experimental setup. The authors gratefully acknowledge the following sources of support. Terry Koo was funded by NSF grant DMS-0434222 and a grant from NTT, Agmt. Dtd. 6/21/1998. Xavier Carreras was supported by the Catalan Ministry of Innovation, Universities and Enterprise, and a grant from NTT, Agmt. Dtd. 6/21/1998. Michael Collins was funded by NSF grants 0347631 and DMS-0434222.

## References

- P.F. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479.
- S. Buchholz and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of CoNLL*, pages 149–164.
- X. Carreras. 2007. Experiments with a Higher-Order Projective Dependency Parser. In *Proceedings of EMNLP-CoNLL*, pages 957–961.
- E. Charniak, D. Blaheta, N. Ge, K. Hall, and M. Johnson. 2000.  *BLLIP 1987–89 WSJ Corpus Release 1, LDC No. LDC2000T43*. Linguistic Data Consortium.
- Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.
- M. Collins, J. Hajič, L. Ramshaw, and C. Tillmann. 1999. A Statistical Parser for Czech. In *Proceedings of ACL*, pages 505–512.
- M. Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of EMNLP*, pages 1–8.
- K. Crammer and Y. Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *Journal of Machine Learning Research*, 3:951–991.
- K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. 2004. Online Passive-Aggressive Algorithms. In S. Thrun, L. Saul, and B. Schölkopf, editors, *NIPS 16*, pages 1229–1236.
- J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.
- J. Eisner. 2000. Bilexical Grammars and Their Cubic-Time Parsing Algorithms. In H. Bunt and A. Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.
- Y. Freund and R. Schapire. 1999. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3):277–296.
- J. Hajič, E. Hajičová, P. Pajas, J. Panevova, and P. Sgall. 2001. *The Prague Dependency Treebank 1.0, LDC No. LDC2001T10*. Linguistics Data Consortium.
- J. Hajič. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In E. Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19.
- K. Hall and V. Novák. 2005. Corrective Modeling for Non-Projective Dependency Parsing. In *Proceedings of IWPT*, pages 42–52.
- T. Koo and M. Collins. 2005. Hidden-Variable Models for Discriminative Reranking. In *Proceedings of HLT-EMNLP*, pages 507–514.
- W. Li and A. McCallum. 2005. Semi-Supervised Sequence Modeling with Syntactic Topic Models. In *Proceedings of AAAI*, pages 813–818.
- P. Liang. 2005. Semi-Supervised Learning for Natural Language. Master’s thesis, Massachusetts Institute of Technology.
- M.P. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with Latent Annotations. In *Proceedings of ACL*, pages 75–82.
- D. McClosky, E. Charniak, and M. Johnson. 2006. Effective Self-Training for Parsing. In *Proceedings of HLT-NAACL*, pages 152–159.
- R. McDonald and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of EACL*, pages 81–88.
- R. McDonald, K. Crammer, and F. Pereira. 2005a. Online Large-Margin Training of Dependency Parsers. In *Proceedings of ACL*, pages 91–98.
- R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005b. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP*, pages 523–530.
- S. Miller, J. Guinness, and A. Zamanian. 2004. Name Tagging with Word Clusters and Discriminative Training. In *Proceedings of HLT-NAACL*, pages 337–342.
- J. Nivre and J. Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of ACL*, pages 99–106.
- J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of EMNLP-CoNLL 2007*, pages 915–932.
- S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of COLING-ACL*, pages 433–440.
- A. Ratnaparkhi. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. In *Proceedings of EMNLP*, pages 133–142.
- I. Titov and J. Henderson. 2007. Constituent Parsing with Incremental Sigmoid Belief Networks. In *Proceedings of ACL*, pages 632–639.
- Q.I. Wang, D. Schuurmans, and D. Lin. 2005. Strictly Lexical Dependency Parsing. In *Proceedings of IWPT*, pages 152–159.
- H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis With Support Vector Machines. In *Proceedings of IWPT*, pages 195–206.

# Optimal $k$ -arization of Synchronous Tree-Adjoining Grammar

**Rebecca Nesson**  
School of Engineering  
and Applied Sciences  
Harvard University  
Cambridge, MA 02138  
nesson@seas.harvard.edu

**Giorgio Satta**  
Department of  
Information Engineering  
University of Padua  
I-35131 Padova, Italy  
satta@dei.unipd.it

**Stuart M. Shieber**  
School of Engineering  
and Applied Sciences  
Harvard University  
Cambridge, MA 02138  
shieber@seas.harvard.edu

## Abstract

Synchronous Tree-Adjoining Grammar (STAG) is a promising formalism for syntax-aware machine translation and simultaneous computation of natural-language syntax and semantics. Current research in both of these areas is actively pursuing its incorporation. However, STAG parsing is known to be NP-hard due to the potential for intertwined correspondences between the linked nonterminal symbols in the elementary structures. Given a particular grammar, the polynomial degree of efficient STAG parsing algorithms depends directly on the *rank* of the grammar: the maximum number of correspondences that appear within a single elementary structure. In this paper we present a compile-time algorithm for transforming a STAG into a strongly-equivalent STAG that optimally minimizes the rank,  $k$ , across the grammar. The algorithm performs in  $\mathcal{O}(|G| + |Y| \cdot L_G^3)$  time where  $L_G$  is the maximum number of links in any single synchronous tree pair in the grammar and  $Y$  is the set of synchronous tree pairs of  $G$ .

## 1 Introduction

Tree-adjoining grammar is a widely used formalism in natural-language processing due to its mildly-context-sensitive expressivity, its ability to naturally capture natural-language argument substitution (via its substitution operation) and optional modification (via its adjunction operation), and the existence of efficient algorithms for processing it. Recently, the desire to incorporate syntax-awareness into machine translation systems has generated interest in

the application of synchronous tree-adjoining grammar (STAG) to this problem (Nesson, Shieber, and Rush, 2006; Chiang and Rambow, 2006). In a parallel development, interest in incorporating semantic computation into the TAG framework has led to the use of STAG for this purpose (Nesson and Shieber, 2007; Han, 2006b; Han, 2006a; Nesson and Shieber, 2006). Although STAG does not increase the expressivity of the underlying formalisms (Shieber, 1994), STAG parsing is known to be NP-hard due to the potential for intertwined correspondences between the linked nonterminal symbols in the elementary structures (Satta, 1992; Weir, 1988). Without efficient algorithms for processing it, its potential for use in machine translation and TAG semantics systems is limited.

Given a particular grammar, the polynomial degree of efficient STAG parsing algorithms depends directly on the *rank* of the grammar: the maximum number of correspondences that appear within a single elementary structure. This is illustrated by the tree pairs given in Figure 1 in which no two numbered links may be isolated. (By “isolated”, we mean that the links can be contained in a fragment of the tree that contains no other links and dominates only one branch not contained in the fragment. A precise definition is given in section 3.)

An analogous problem has long been known to exist for synchronous context-free grammars (SCFG) (Aho and Ullman, 1969). The task of producing efficient parsers for SCFG has recently been addressed by binarization or  $k$ -arization of SCFG grammars that produce equivalent grammars in which the rank,  $k$ , has been minimized (Zhang



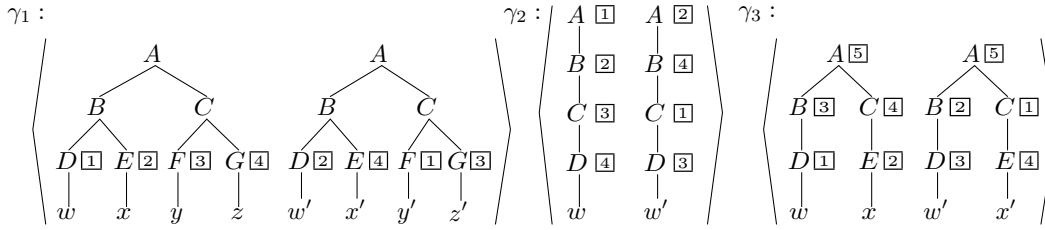


Figure 1: Example of intertwined links that cannot be binarized. No two links can be isolated in both trees in a tree pair. Note that in tree pair  $\gamma_1$ , any set of three links may be isolated while in tree pair  $\gamma_2$ , no group of fewer than four links may be isolated. In  $\gamma_3$  no group of links smaller than four may be isolated.

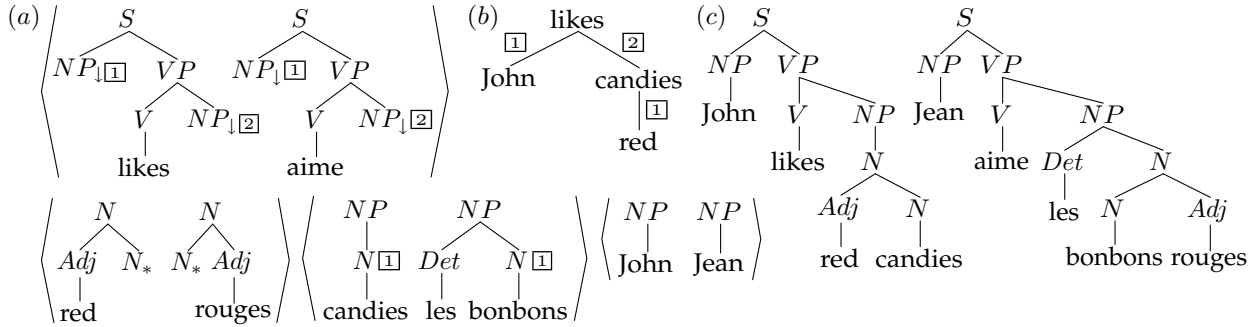


Figure 2: An example STAG derivation of the English/French sentence pair “John likes red candies”/“Jean aime les bonbons rouges”. The figure is divided as follows: (a) the STAG grammar, (b) the derivation tree for the sentence pair, and (c) the derived tree pair for the sentences.

and Gildea, 2007; Zhang et al., 2006; Gildea, Satta, and Zhang, 2006). The methods for  $k$ -arization of SCFG cannot be directly applied to STAG because of the additional complexity introduced by the expressivity-increasing adjunction operation of TAG. In SCFG, where substitution is the only available operation and the depth of elementary structures is limited to one, the  $k$ -arization problem reduces to analysis of permutations of strings of non-terminal symbols. In STAG, however, the arbitrary depth of the elementary structures and the lack of restriction to contiguous strings of nonterminals introduced by adjunction substantially complicate the task.

In this paper we offer the first algorithm addressing this problem for the STAG case. We present a compile-time algorithm for transforming a STAG into a strongly-equivalent STAG that optimally minimizes  $k$  across the grammar. This is a critical minimization because  $k$  is the feature of the grammar that appears in the exponent of the complexity of parsing algorithms for STAG. Following the method of Seki

et al. (1991), an STAG parser can be implemented with complexity  $\mathcal{O}(n^{4 \cdot (k+1)} \cdot |G|)$ . By minimizing  $k$ , the worst-case complexity of a parser instantiated for a particular grammar is optimized. The  $k$ -arization algorithm performs in  $\mathcal{O}(|G| + |Y| \cdot L_G^3)$  time where  $L_G$  is the maximum number of links in any single synchronous tree pair in the grammar and  $Y$  is the set of synchronous tree pairs of  $G$ . By comparison, a baseline algorithm performing exhaustive search requires  $\mathcal{O}(|G| + |Y| \cdot L_G^6)$  time.<sup>1</sup>

The remainder of the paper proceeds as follows. In section 2 we provide a brief introduction to the STAG formalism. We present the  $k$ -arization algorithm in section 3 and an analysis of its complexity in section 4. We prove the correctness of the algorithm in section 5.

<sup>1</sup>In a synchronous tree pair with  $L$  links, there are  $\mathcal{O}(L^4)$  pairs of valid fragments. It takes  $\mathcal{O}(L)$  time to check if the two components in a pair have the same set of links. Once the synchronous fragment with the smallest number of links is excised, this process iterates at most  $L$  times, resulting in time  $\mathcal{O}(L_G^6)$ .

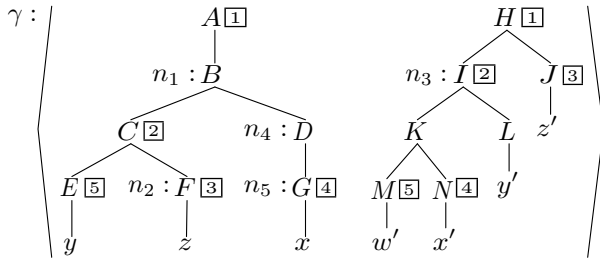


Figure 3: A synchronous tree pair containing fragments  $\alpha_L = \gamma_L(n_1, n_2)$  and  $\alpha_R = \gamma_R(n_3)$ . Since  $\text{links}(n_1, n_2) = \text{links}(n_3) = \{\underline{2}, \underline{4}, \underline{5}\}$ , we can define synchronous fragment  $\alpha = \langle \alpha_L, \alpha_R \rangle$ . Note also that node  $n_3$  is a maximal node and node  $n_5$  is not.  $\sigma(n_1) = \underline{2}\underline{5}\underline{5}\underline{3}\underline{3}\underline{2}\underline{4}\underline{4}$ ;  $\sigma(n_3) = \underline{2}\underline{5}\underline{5}\underline{4}\underline{4}\underline{2}$ .

## 2 Synchronous Tree-Adjoining Grammar

A tree-adjoining grammar (TAG) consists of a set of elementary tree structures of arbitrary depth, which are combined by substitution, familiar from context-free grammars, or an operation of adjunction that is particular to the TAG formalism. **Auxiliary trees** are elementary trees in which the root and a frontier node, called the **foot node** and distinguished by the diacritic \*, are labeled with the same nonterminal  $A$ . The adjunction operation involves splicing an auxiliary tree in at an internal node in an elementary tree also labeled with nonterminal  $A$ . Trees without a foot node, which serve as a base for derivations, are called **initial trees**. For further background, refer to the survey by Joshi and Schabes (1997).

We depart from the traditional definition in notation only by specifying adjunction and substitution sites explicitly with numbered links. Each link may be used only once in a derivation. Operations may only occur at nodes marked with a link. For simplicity of presentation we provisionally assume that only one link is permitted at a node. We later drop this assumption.

In a synchronous TAG (STAG) the elementary structures are ordered pairs of TAG trees, with a linking relation specified over pairs of nonterminal nodes. Each link has two locations, one in the left tree in a pair and the other in the right tree. An example of an STAG derivation including both substitution and adjunction is given in Figure 2. For further background, refer to the work of Shieber and Schabes (1990) and Shieber (1994).

## 3 $k$ -arization Algorithm

For a synchronous tree pair  $\gamma = \langle \gamma_L, \gamma_R \rangle$ , a **fragment** of  $\gamma_L$  (or  $\gamma_R$ ) is a complete subtree rooted at some node  $n$  of  $\gamma_L$ , written  $\gamma_L(n)$ , or else a subtree rooted at  $n$  with a gap at node  $n'$ , written  $\gamma_L(n, n')$ ; see Figure 3 for an example. We write  $\text{links}(n)$  and  $\text{links}(n, n')$  to denote the set of links of  $\gamma_L(n)$  and  $\gamma_L(n, n')$ , respectively. When we do not know the root or gap nodes of some fragment  $\alpha_L$ , we also write  $\text{links}(\alpha_L)$ .

We say that a set of links  $\Lambda$  from  $\gamma$  can be **isolated** if there exist fragments  $\alpha_L$  and  $\alpha_R$  of  $\gamma_L$  and  $\gamma_R$ , respectively, both with links  $\Lambda$ . If this is the case, we can construct a synchronous fragment  $\alpha = \langle \alpha_L, \alpha_R \rangle$ . The goal of our algorithm is to decompose  $\gamma$  into synchronous fragments such that the maximum number of links of a synchronous fragment is kept to a minimum, and  $\gamma$  can be obtained from the synchronous fragments by means of the usual substitution and adjunction operations. In order to simplify the presentation of our algorithm we assume, without any loss of generality, that all elementary trees of the source STAG have nodes with at most two children.

### 3.1 Maximal Nodes

A node  $n$  of  $\gamma_L$  (or  $\gamma_R$ ) is called **maximal** if (i)  $\text{links}(n) \neq \emptyset$ , and (ii) it is either the root node of  $\gamma_L$  or, for its parent node  $n'$ , we have  $\text{links}(n') \neq \text{links}(n)$ . Note that for every node  $n'$  of  $\gamma_L$  such that  $\text{links}(n') \neq \emptyset$  there is always a unique maximal node  $n$  such that  $\text{links}(n') = \text{links}(n)$ . Thus, for the purpose of our algorithm, we need only look at maximal nodes as places for excising tree fragments. We can show that the number of maximal nodes  $M_n$  in a subtree  $\gamma_L(n)$  always satisfies  $|\text{links}(n)| \leq M_n \leq 2 \times |\text{links}(n)| - 1$ .

Let  $n$  be some node of  $\gamma_L$ , and let  $l(n)$  be the (unique) link impinging on  $n$  if such a link exists, and  $l(n) = \varepsilon$  otherwise. We associate  $n$  with a string  $\sigma(n)$ , defined by a pre- and post-order traversal of fragment  $\gamma_L(n)$ . The symbols of  $\sigma(n)$  are the links in  $\text{links}(n)$ , viewed as atomic symbols. Given a node  $n$  with  $p$  children  $n_1, \dots, n_p$ ,  $0 \leq p \leq 2$ , we define  $\sigma(n) = l(n) \sigma(n_1) \cdots \sigma(n_p) l(n)$ . See again Figure 3 for an example. Note that  $|\sigma(n)| = 2 \times |\text{links}(n)|$ .

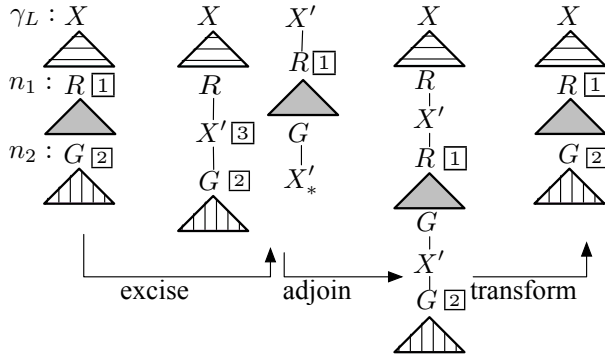


Figure 4: A diagram of the tree transformation performed when fragment  $\gamma_L(n_1, n_2)$  is removed. In this and the diagrams that follow, patterned or shaded triangles represent segments of the tree that contain multiple nodes and at least one link. Where the pattern or shading corresponds across trees in a tree pair, the set of links contained within those triangles are equivalent.

### 3.2 Excision of Synchronous Fragments

Although it would be possible to excise synchronous fragments without creating new nonterminal nodes, for clarity we present a simple tree transformation when a fragment is excised that leaves existing nodes intact. A schematic depiction is given in Figure 4. In the figure, we demonstrate the excision process on one half of a synchronous fragment:  $\gamma_L(n_1, n_2)$  is excised to form two new trees. The excised tree is not processed further. In the excision process the root and gap nodes of the original tree are not altered. The material between them is replaced with a single new node with a fresh nonterminal symbol and a fresh link number. This nonterminal node and link form the adjunction or substitution site for the excised tree. Note that any link impinging on the root node of the excised fragment is by our convention included in the fragment and any link impinging on the gap node is not.

To regenerate the original tree, the excised fragment can be adjoined or substituted back into the tree from which it was excised. The new nodes that were generated in the excision may be removed and the original root and gap nodes may be merged back together retaining any impinging links, respectively. Note that if there was a link on either the root or gap node in the original tree, it is not lost or duplicated

1	1	0	0	0	0	0	0	0	0	1
2	0	1	0	0	0	0	1	0	0	0
5	0	0	1	1	0	0	0	0	0	0
5	0	0	1	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	1	0
3	0	0	0	0	0	0	0	1	1	0
2	0	1	0	0	0	0	1	0	0	0
4	0	0	0	0	1	1	0	0	0	0
4	0	0	0	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	1
	1	2	5	5	4	4	2	3	3	1

Figure 5: Table  $\pi$  with synchronous fragment  $\langle \gamma_L(n_1, n_2), \gamma_R(n_3) \rangle$  from Figure 3 highlighted.

in the process.

### 3.3 Method

Let  $n_L$  and  $n_R$  be the root nodes of trees  $\gamma_L$  and  $\gamma_R$ , respectively. We know that  $\text{links}(n_L) = \text{links}(n_R)$ , and  $|\sigma(n_L)| = |\sigma(n_R)|$ , the second string being a rearrangement of the occurrences of symbols in the first one. The main data structure of our algorithm is a Boolean matrix  $\pi$  of size  $|\sigma(n_L)| \times |\sigma(n_L)|$ , whose rows are addressed by the occurrences of symbols in  $\sigma(n_L)$ , in the given order, and whose columns are similarly addressed by  $\sigma(n_R)$ . For occurrences of links  $\overline{x_1}, \overline{x_2}$ , the element of  $\pi$  at a row addressed by  $x_1$  and a column addressed by  $x_2$  is 1 if  $x_1 = x_2$ , and 0 otherwise. Thus, each row and column of  $\pi$  has exactly two non-zero entries. See Figure 5 for an example.

For a maximal node  $n_1$  of  $\gamma_L$ , we let  $\pi(n_1)$  denote the stripe of adjacent rows of  $\pi$  addressed by substring  $\sigma(n_1)$  of  $\sigma(n_L)$ . If  $n_1$  dominates  $n_2$  in  $\gamma_L$ , we let  $\pi(n_1, n_2)$  denote the rows of  $\pi$  addressed by  $\sigma(n_1)$  but not by  $\sigma(n_2)$ . This forms a pair of horizontal stripes in  $\pi$ . For nodes  $n_3, n_4$  of  $\gamma_R$ , we similarly define  $\pi(n_3)$  and  $\pi(n_3, n_4)$  as vertical stripes of adjacent columns. See again Figure 5.

Our algorithm is reported in Figure 6. For each synchronous tree pair  $\gamma = \langle \gamma_L, \gamma_R \rangle$  from the input grammar, we maintain an agenda  $B$  with all candidate fragments  $\alpha_L$  from  $\gamma_L$  having at least two links. These fragments are processed greedily in order of increasing number of links. The function ISOLATE(), described in more detail be-

```

1: Function KARIZE( $G$ ) { $G$  a binary STAG}
2:  $G' \leftarrow$  STAG with empty set of synch trees;
3: for all  $\gamma = \langle \gamma_L, \gamma_R \rangle$  in  $G$  do
4:   init  $\pi$  and  $B$ ;
5:   while  $B \neq \emptyset$  do
6:      $\alpha_L \leftarrow$  next fragment from  $B$ ;
7:      $\alpha_R \leftarrow$  ISOLATE( $\alpha_L, \pi, \gamma_R$ );
8:     if  $\alpha_R \neq \text{null}$  then
9:       add  $\langle \alpha_L, \alpha_R \rangle$  to  $G'$ ;
10:       $\gamma \leftarrow$  excise  $\langle \alpha_L, \alpha_R \rangle$  from  $\gamma$ ;
11:      update  $\pi$  and  $B$ ;
12:   add  $\gamma$  to  $G'$ ;
13: return  $G'$ 

```

Figure 6: Main algorithm.

low, looks for a right fragment  $\alpha_R$  with the same links as  $\alpha_L$ . Upon success, the synchronous fragment  $\alpha = \langle \alpha_L, \alpha_R \rangle$  is added to the output grammar. Furthermore, we excise  $\alpha$  from  $\gamma$  and update data structures  $\pi$  and  $B$ . The above process is iterated until  $B$  becomes empty. We show in section 5 that this greedy strategy is sound and complete.

The function ISOLATE() is specified in Figure 7. We take as input a left fragment  $\alpha_L$ , which is associated with one or two horizontal stripes in  $\pi$ , depending on whether  $\alpha_L$  has a gap node or not. The left boundary of  $\alpha_L$  in  $\pi$  is the index  $x_1$  of the column containing the leftmost occurrence of a 1 in the horizontal stripes associated with  $\alpha_L$ . Similarly, the right boundary of  $\alpha_L$  in  $\pi$  is the index  $x_2$  of the column containing the rightmost occurrence of a 1 in these stripes. We retrieve the shortest substring  $\sigma(n)$  of  $\sigma(n_R)$  that spans over indices  $x_1$  and  $x_2$ . This means that  $n$  is the lowest node from  $\gamma_R$  such that the links of  $\alpha_L$  are a subset of the links of  $\gamma_R(n)$ .

If the condition at line 3 is satisfied, all of the matrix entries of value 1 that are found from column  $x_1$  to column  $x_2$  fall within the horizontal stripes associated with  $\alpha_L$ . In this case we can report the right fragment  $\alpha_R = \gamma_R(n)$ . Otherwise, we check whether the entries of value 1 that fall outside of the two horizontal stripes in between columns  $x_1$  and  $x_2$  occur within adjacent columns, say from column  $x_3 \geq x_1$  to column  $x_4 \leq x_2$ . In this case, we check whether there exists some node  $n'$  such that the substring of  $\sigma(n)$  from position  $x_3$  to  $x_4$  is

```

1: Function ISOLATE( $\alpha_L, \pi, \gamma_R$ )
2: select  $n \in \gamma_R$  such that  $\sigma(n)$  is the shortest
   string within  $\sigma(n_R)$  including left/right bound-
   aries of  $\alpha_L$  in  $\pi$ ;
3: if  $|\sigma(n)| = 2 \times |\text{links}(\alpha_L)|$  then
4:   return  $\gamma_R(n)$ ;
5: select  $n' \in \gamma_R$  such that  $\sigma(n')$  is the gap string
   within  $\sigma(n)$  for which  $\text{links}(n) - \text{links}(n') =$ 
    $\text{links}(\alpha_L)$ ;
6: if  $n'$  is not defined then
7:   return null; {more than one gap}
8: return  $\gamma_R(n, n')$ ;

```

Figure 7: Find synchronous fragment.

an occurrence of string  $\sigma(n')$ . This means that  $n'$  is the gap node, and we report the right fragment  $\alpha_R = \gamma_R(n, n')$ . See again Figure 5.

We now drop the assumption that only one link may impinge on a node. When multiple links impinge on a single node  $n$ ,  $l(n)$  is an arbitrary order over those links. In the execution of the algorithm, any stripe that contains one link in  $l(n)$  it must include every link in  $l(n)$ . This prevents the excision of a proper subset of the links at any node. This preserves correctness because excising any proper subset would impose an order over the links at  $n$  that is not enforced in the input grammar. Because the links at a node are treated as a unit, the complexity of the algorithm is not affected.

## 4 Complexity

We discuss here an implementation of the algorithm of section 3 resulting in time complexity  $\mathcal{O}(|G| + |Y| \cdot L_G^3)$ , where  $Y$  is the set of synchronous tree pairs of  $G$  and  $L_G$  is the maximum number of links in a synchronous tree pair in  $Y$ .

Consider a synchronous tree pair  $\gamma = \langle \gamma_L, \gamma_R \rangle$  with  $L$  links. If  $M$  is the number of maximal nodes in  $\gamma_L$  or  $\gamma_R$ , we have  $M = \Theta(L)$  (Section 3.1). We implement the sparse table  $\pi$  in  $\mathcal{O}(L)$  space, recording for each row and column the indices of its two non-zero entries. We also assume that we can go back and forth between maximal nodes  $n$  and strings  $\sigma(n)$  in constant time. Here each  $\sigma(n)$  is represented by its boundary positions within  $\sigma(n_L)$  or  $\sigma(n_R)$ ,  $n_L$  and  $n_R$  the root nodes of  $\gamma_L$  and  $\gamma_R$ , respectively.

At line 2 of the function ISOLATE() (Figure 7) we retrieve the left and right boundaries by scanning the rows of  $\pi$  associated with input fragment  $\alpha_L$ . We then retrieve node  $n$  by visiting all maximal nodes of  $\gamma_L$  spanning these boundaries. Under the above assumptions, this can be done in time  $\mathcal{O}(L)$ . In a similar way we can implement line 5, resulting in overall run time  $\mathcal{O}(L)$  for function ISOLATE().

In the function KARIZE() (Figure 6) we use buckets  $B_i$ ,  $1 \leq i \leq L$ , where each  $B_i$  stores the candidate fragments  $\alpha_L$  with  $|\text{links}(\alpha_L)| = i$ . To populate these buckets, we first process fragments  $\gamma_L(n)$  by visiting bottom up the maximal nodes of  $\gamma_L$ . The quantity  $|\text{links}(n)|$  is computed from the quantities  $|\text{links}(n_i)|$ , where  $n_i$  are the highest maximal nodes dominated by  $n$ . (There are at most two such nodes.) Fragments  $\gamma_L(n, n')$  can then be processed using the relation  $|\text{links}(n, n')| = |\text{links}(n)| - |\text{links}(n')|$ . In this way each fragment is processed in constant time, and population of all the buckets takes  $\mathcal{O}(L^2)$  time.

We now consider the while loop at lines 5 to 11 in function KARIZE(). For a synchronous tree pair  $\gamma$ , the loop iterates once for each candidate fragment  $\alpha_L$  in some bucket. We have a total of  $\mathcal{O}(L^2)$  iterations, since the initial number of candidates in the buckets is  $\mathcal{O}(L^2)$ , and the possible updating of the buckets after a synchronous fragment is removed does not increase the total size of all the buckets. If the links in  $\alpha_L$  cannot be isolated, one iteration takes time  $\mathcal{O}(L)$  (the call to function ISOLATE()). If the links in  $\alpha_L$  can be isolated, then we need to restructure  $\pi$  and to repopulate the buckets. The former can be done in time  $\mathcal{O}(L)$  and the latter takes time  $\mathcal{O}(L^2)$ , as already discussed. Crucially, the updating of  $\pi$  and the buckets takes place no more than  $L - 1$  times. This is because each time we excise a synchronous fragment, the number of links in  $\gamma$  is reduced by at least one.

We conclude that function KARIZE() takes time  $\mathcal{O}(L^3)$  for each synchronous tree  $\gamma$ , and the total running time is  $\mathcal{O}(|G| + |Y| \cdot L_G^3)$ , where  $Y$  is the set of synchronous tree pairs of  $G$ . The term  $|G|$  accounts for the reading of the input, and dominates the complexity of the algorithm only in case there are very few links in each synchronous tree pair.

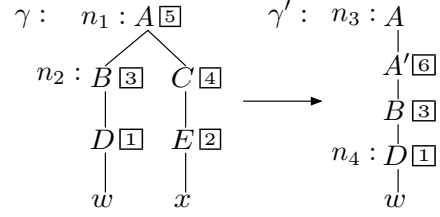


Figure 8: In  $\gamma$  links  $\boxed{3}$  and  $\boxed{5}$  cannot be isolated because the fragment would have to contain two gaps. However, after the removal of fragment  $\gamma(n_1, n_2)$ , an analogous fragment  $\gamma'(n_3, n_4)$  may be removed.

## 5 Proof of Correctness

The algorithm presented in the previous sections produces an optimal  $k$ -arization for the input grammar. In this section we sketch a proof of correctness of the strategy employed by the algorithm.<sup>2</sup>

The  $k$ -arization strategy presented above is greedy in that it always chooses the excisable fragment with the smallest number of links at each step and does not perform any backtracking. We must therefore show that this process cannot result in a non-optimal solution. If fragments could not overlap each other, this would be trivial to show because the excision process would be confluent. If all overlapping fragments were cases of complete containment of one fragment within another, the proof would also be trivial because the smallest-to-largest excision order would guarantee optimality. However, it is possible for fragments to partially overlap each other, meaning that the intersection of the set of links contained in the two fragments is non-empty and the difference between the set of links in one fragment and the other is also non-empty. Overlapping fragment configurations are given in Figure 9 and discussed in detail below.

The existence of partially overlapping fragments complicates the proof of optimality for two reasons. First, the excision of a fragment  $\alpha$  that is partially overlapped with another fragment  $\beta$  necessarily precludes the excision of  $\beta$  at a later stage in the ex-

<sup>2</sup>Note that the soundness of the algorithm can be easily verified from the fact that the removal of fragments can be reversed by performing standard STAG adjunction and substitution operations until a single STAG tree pair is produced. This tree pair is trivially homomorphic to the original tree pair and can easily be mapped to the original tree pair.

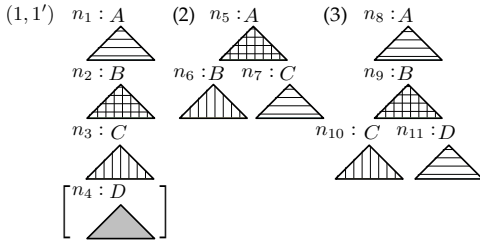


Figure 9: The four possible configurations of overlapped fragments within a single tree. For type 1, let  $\alpha = \gamma(n_1, n_3)$  and  $\beta = \gamma(n_2, n_4)$ . The roots and gaps of the fragments are interleaved. For type 1', let  $\alpha = \gamma(n_1, n_3)$  and  $\beta = \gamma(n_2)$ . The root of  $\beta$  dominates the gap of  $\alpha$ . For type 2, let  $\alpha = \gamma(n_5, n_6)$  and  $\beta = \gamma(n_5, n_7)$ . The fragments share a root and have gap nodes that do not dominate each other. For type 3 let  $\alpha = \gamma(n_8, n_{10})$  and  $\beta = \gamma(n_9, n_{11})$ . The root of  $\alpha$  dominates the root of  $\beta$ , both roots dominate both gaps, but neither gap dominates the other.

cision process. Second, the removal of a fragment may cause a previously non-isolatable set of links to become isolatable, effectively creating a new fragment that may be advantageous to remove. This is demonstrated in Figure 8. These possibilities raise the question of whether the choice between removing fragments  $\alpha$  and  $\beta$  may have consequences at a later stage in the excision process. We demonstrate that this choice cannot affect the  $k$  found for a given grammar.

We begin by sketching the proof of a lemma that shows that removal of a fragment  $\beta$  that partially overlaps another fragment  $\alpha$  always leaves an analogous fragment that may be removed.

### 5.1 Validity Preservation

Consider a STAG tree pair  $\gamma$  containing the set of links  $\Lambda$  and two synchronous fragments  $\alpha$  and  $\beta$  with  $\alpha$  containing links  $\text{links}(\alpha)$  and  $\beta$  containing  $\text{links}(\beta)$  ( $\text{links}(\alpha), \text{links}(\beta) \subseteq \Lambda$ ).

If  $\alpha$  and  $\beta$  do not overlap, the removal of  $\beta$  is defined as validity preserving with respect to  $\alpha$ .

If  $\alpha$  and  $\beta$  overlap, removal of  $\beta$  from  $\gamma$  is **validity preserving** with respect to  $\alpha$  if after the removal there exists a valid synchronous fragment (containing at most one gap on each side) that contains all and only the links  $(\text{links}(\alpha) - \text{links}(\beta)) \cup \{\boxed{\alpha}\}$  where  $\boxed{\alpha}$  is the new link added to  $\gamma$ .

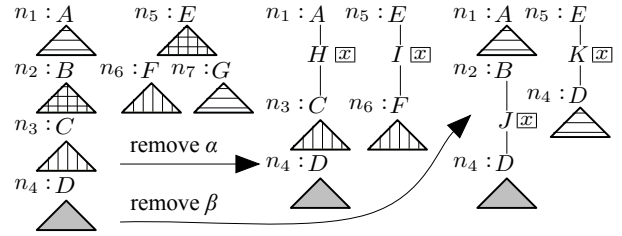


Figure 10: Removal from a tree pair  $\gamma$  containing type 1–type 2 fragment overlap. The fragment  $\alpha$  is represented by the horizontal-lined pieces of the tree pair. The fragment  $\beta$  is represented by the vertical-lined pieces of the tree pair. Cross-hatching indicates the overlapping portion of the two fragments.

We prove a lemma that removal of any synchronous fragment from an STAG tree pair is validity preserving with respect to all of the other synchronous fragments in the tree pair.

It suffices to show that for two arbitrary synchronous fragments  $\alpha$  and  $\beta$ , the removal of  $\beta$  is validity preserving with respect to  $\alpha$ . We show this by examination of the possible configurations of  $\alpha$  and  $\beta$ .

Consider the case in which  $\beta$  is fully contained within  $\alpha$ . In this case  $\text{links}(\beta) \subsetneq \text{links}(\alpha)$ . The removal of  $\beta$  leaves the root and gap of  $\alpha$  intact in both trees in the pair, so it remains a valid fragment. The new link is added at the new node inserted where  $\beta$  was removed. Since  $\beta$  is fully contained within  $\alpha$ , this node is below the root of  $\alpha$  but not below its gap. Thus, the removal process leaves  $\alpha$  with the links  $(\text{links}(\alpha) - \text{links}(\beta)) \cup \{\boxed{\alpha}\}$ , where  $\boxed{\alpha}$  is the link added in the removal process; the removal is validity preserving.

Synchronous fragments may partially overlap in several different ways. There are four possible configurations for an overlapped fragment within a single tree, depicted in Figure 9. These different single-tree overlap types can be combined in any way to form valid synchronous fragments. Due to space constraints, we consider two illustrative cases and leave the remainder as an exercise to the reader.

An example of removing fragments from a tree set containing type 1–type 2 overlapped fragments is given in Figure 10. Let  $\alpha = \langle \gamma_L(n_1, n_3), \gamma_R(n_5, n_6) \rangle$ . Let

$\beta = \langle \gamma_L(n_2, n_4), \gamma_R(n_5, n_7) \rangle$ . If  $\alpha$  is removed, the validity preserving fragment for  $\beta$  is  $\langle \gamma'_L(n_1, n_4), \gamma'_R(n_5) \rangle$ . It contains the links in the vertical-lined part of the tree and the new link  $\boxed{x}$ . This forms a valid fragment because both sides contain at most one gap and both contain the same set of links. In addition, it is validity preserving for  $\beta$  because it contains exactly the set of links that were in  $\text{links}(\beta)$  and not in  $\text{links}(\alpha)$  plus the new link  $\boxed{x}$ . If we instead choose to remove  $\beta$ , the validity preserving fragment for  $\alpha$  is  $\langle \gamma'_L(n_1, n_4), \gamma'_R(n_5) \rangle$ . The links in each side of this fragment are the same, each side contains at most one gap, and the set of links is exactly the set left over from  $\text{links}(\alpha)$  once  $\text{links}(\beta)$  is removed plus the newly generated link  $\boxed{x}$ .

An example of removing fragments from a tree set containing type 1'–type 3 (reversed) overlapped fragments is given in Figure 11. If  $\alpha$  is removed, the validity preserving fragment for  $\beta$  is  $\langle \gamma'_L(n_1), \gamma'_R(n_4) \rangle$ . If  $\beta$  is removed, the validity preserving fragment for  $\alpha$  is  $\langle \gamma'_L(n_1, n_8), \gamma'_R(n_4) \rangle$ .

Similar reasoning follows for all remaining types of overlapped fragments.

## 5.2 Proof Sketch

We show that smallest-first removal of fragments is optimal. Consider a decision point at which a choice is made about which fragment to remove. Call the size of the smallest fragments at this point  $m$ , and let the set of fragments of size  $m$  be  $X$  with  $\alpha, \beta \in X$ .

There are two cases to consider. First, consider two partially overlapped fragments  $\alpha \in X$  and  $\delta \notin X$ . Note that  $|\text{links}(\alpha)| < |\text{links}(\delta)|$ . Validity preservation of  $\alpha$  with respect to  $\delta$  guarantees that  $\delta$  or its validity preserving analog will still be available for excision after  $\alpha$  is removed. Excising  $\delta$  increases  $k$  more than excising  $\alpha$  or any fragment that removal of  $\alpha$  will lead to before  $\delta$  is considered. Thus, removal of  $\delta$  cannot result in a smaller value for  $k$  if it is removed before  $\alpha$  rather than after  $\alpha$ .

Second, consider two partially overlapped fragments  $\alpha, \beta \in X$ . Due to the validity preservation lemma, we may choose arbitrarily between the fragments in  $X$  without jeopardizing our ability to later remove other fragments (or their validity preserving analogs) in that set. Removal of fragment  $\alpha$  cannot increase the size of any remaining fragment.

Removal of  $\alpha$  or  $\beta$  may generate new fragments

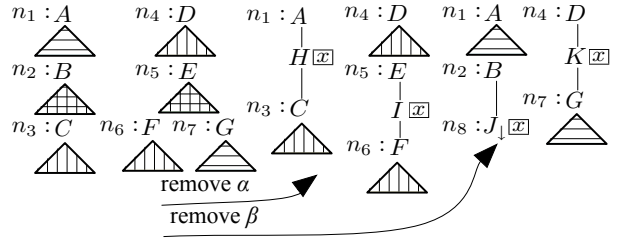


Figure 11: Removal from a tree pair  $\gamma$  containing a type 1'–type 3 (reversed) fragment overlap. The fragment  $\alpha$  is represented by the horizontal lined pieces of the tree pair. The fragment  $\beta$  is represented by the vertical-lined pieces of the tree pair. Cross-hatching indicates the overlapping portion of the two fragments.

that were not previously valid and may reduce the size of existing fragments that it overlaps. In addition, removal of  $\alpha$  may lead to availability of smaller fragments at the next removal step than removal of  $\beta$  (and vice versa). However, since removal of either  $\alpha$  or  $\beta$  produces a  $k$  of size at least  $m$ , the later removal of fragments of size less than  $m$  cannot affect the  $k$  found by the algorithm. Due to validity preservation, removal of any of these smaller fragments will still permit removal of all currently existing fragments or their analogs at a later step in the removal process.

If the removal of  $\alpha$  generates a new fragment  $\delta$  of size larger than  $m$  all remaining fragments in  $X$  (and all others smaller than  $\delta$ ) will be removed before  $\delta$  is considered. Therefore, if removal of  $\beta$  generates a new fragment smaller than  $\delta$ , the smallest-first strategy will properly guarantee its removal before  $\delta$ .

## 6 Conclusion

In order for STAG to be used in machine translation and other natural-language processing tasks it must be possible to process it efficiently. The difficulty in parsing STAG stems directly from the factor  $k$  that indicates the degree to which the correspondences are intertwined within the elementary structures of the grammar. The algorithm presented in this paper is the first method available for  $k$ -arizing a synchronous TAG grammar into an equivalent grammar with an optimal value for  $k$ . The algorithm operates offline and requires only  $\mathcal{O}(|G| + |Y| \cdot L_G^3)$  time. Both the derivation trees and derived trees produced are trivially homomorphic to those that are produced by the original grammar.

## References

- Aho, Alfred V. and Jeffrey D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56.
- Chiang, David and Owen Rambow. 2006. The hidden TAG model: synchronous grammars for parsing resource-poor languages. In *Proceedings of the 8th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 8)*, pages 1–8.
- Gildea, Daniel, Giorgio Satta, and Hao Zhang. 2006. Factoring synchronous grammars by sorting. In *Proceedings of the International Conference on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL-06)*, July.
- Han, Chung-Hye. 2006a. Pied-piping in relative clauses: Syntax and compositional semantics based on synchronous tree adjoining grammar. In *Proceedings of the 8th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 8)*, pages 41–48, Sydney, Australia.
- Han, Chung-Hye. 2006b. A tree adjoining grammar analysis of the syntax and semantics of it-clefts. In *Proceedings of the 8th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 8)*, pages 33–40, Sydney, Australia.
- Joshi, Aravind K. and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*. Springer, pages 69–124.
- Nesson, Rebecca and Stuart M. Shieber. 2006. Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*, Malaga, Spain, 29–30 July.
- Nesson, Rebecca and Stuart M. Shieber. 2007. Extraction phenomena in synchronous TAG syntax and semantics. In *Proceedings of Syntax and Structure in Statistical Translation (SSST)*, Rochester, NY, April.
- Nesson, Rebecca, Stuart M. Shieber, and Alexander Rush. 2006. Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA 2006)*, Boston, Massachusetts, 8-12 August.
- Satta, Giorgio. 1992. Recognition of linear context-free rewriting systems. In *Proceedings of the 10th Meeting of the Association for Computational Linguistics (ACL92)*, pages 89–95, Newark, Delaware.
- Seki, H., T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.
- Shieber, Stuart M. 1994. Restricting the weak-generative capacity of synchronous tree-adjoining grammars. *Computational Intelligence*, 10(4):371–385, November.
- Shieber, Stuart M. and Yves Schabes. 1990. Synchronous tree adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING '90)*, Helsinki, August.
- Weir, David. 1988. Characterizing mildly context-sensitive grammar formalisms. PhD Thesis, Department of Computer and Information Science, University of Pennsylvania.
- Zhang, Hao and Daniel Gildea. 2007. Factorization of synchronous context-free grammars in linear time. In *NAACL Workshop on Syntax and Structure in Statistical Translation (SSST)*, April.
- Zhang, Hao, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.



# Enhancing Performance of Lexicalised Grammars

Rebecca Dridan<sup>†</sup>, Valia Kordoni<sup>†</sup>, Jeremy Nicholson<sup>†‡</sup>

<sup>†</sup>Dept of Computational Linguistics, Saarland University and DFKI GmbH, Germany

<sup>‡</sup>Dept of Computer Science and Software Engineering and NICTA, University of Melbourne, Australia

{rdrid, kordoni}@coli.uni-sb.de, jeremymn@csse.unimelb.edu.au

## Abstract

This paper describes how external resources can be used to improve parser performance for heavily lexicalised grammars, looking at both robustness and efficiency. In terms of robustness, we try using different types of external data to increase lexical coverage, and find that simple POS tags have the most effect, increasing coverage on unseen data by up to 45%. We also show that filtering lexical items in a supertagging manner is very effective in increasing efficiency. Even using vanilla POS tags we achieve some efficiency gains, but when using detailed lexical types as supertags we manage to halve parsing time with minimal loss of coverage or precision.

## 1 Introduction

Heavily lexicalised grammars have been used in applications such as machine translation and information extraction because they can produce semantic structures which provide more information than less informed parsers. In particular, because of the structural and semantic information attached to lexicon items, these grammars do well at describing complex relationships, like non-projectivity and center embedding. However, the cost of this additional information sometimes makes deep parsers that use these grammars impractical. Firstly because, if the information is not available, the parsers may fail to produce an analysis, a failure of robustness. Secondly, the effect of analysing the extra information can slow the parser down, causing efficiency problems. This paper describes experiments aimed at

improving parser performance in these two areas, by annotating the input given to one such deep parser, the PET parser (Callmeier, 2000), which uses lexicalised grammars developed under the HPSG formalism (Pollard and Sag, 1994).

## 2 Background

In all heavily lexicalised formalisms, such as LTAG, CCG, LFG and HPSG, the lexicon plays a key role in parsing. But a lexicon can never hope to contain all words in open domain text, and so lexical coverage is a central issue in boosting parser robustness. Some systems use heuristics based on numbers, capitalisation and perhaps morphology to guess the category of the unknown word (van Noord and Malouf, 2004), while others have focused on automatically expanding the lexicon (Baldwin, 2005; Hockenmaier et al., 2002; O'Donovan et al., 2005). Another method, described in Section 4, uses external resources such as part-of-speech (POS) tags to select generic lexical entries for out-of-vocabulary words. In all cases, we lose some of the depth of information the hand-crafted lexicon would provide, but an analysis is still produced, though possibly less than fully specified.

The central position of these detailed lexicons causes problems, not only of robustness, but also of efficiency and ambiguity. Many words may have five, six or more lexicon entries associated with them, and this can lead to an enormous search space for the parser. Various means of filtering this search space have been attempted. Kiefer et al. (1999) describes a method of filtering lexical items by specifying and checking for required prefixes and particles

which is particularly effective for German, but also applicable to English. Other research has looked at using dependencies to restrict the parsing process (Sagae et al., 2007), but the most well known filtering method is supertagging. Originally described by Bangalore and Joshi (1994) for use in LTAG parsing, it has also been used very successfully for CCG (Clark, 2002). Supertagging is the process of assigning probable ‘supertags’ to words before parsing to restrict parser ambiguity, where a supertag is a tag that includes more specific information than the typical POS tags. The supertags used in each formalism differ, being elementary trees in LTAG and CCG categories for CCG. Section 3.2 describes an experiment akin to supertagging for HPSG, where the supertags are HPSG lexical types. Unlike elementary trees and CCG categories, which are predominantly syntactic categories, the HPSG lexical types contain a lot of semantic information, as well as syntactic.

In the case study we describe here, the tools, grammars and treebanks we use are taken from work carried out in the DELPH-IN<sup>1</sup> collaboration. This research is based on using HPSG along with Minimal Recursion Semantics (MRS: Copestake et al. (2001)) as a platform to develop deep natural language processing tools, with a focus on multilinguality. The grammars are designed to be bidirectional (used for generation as well as parsing) and so contain very specific linguistic information. In this work, we focus on techniques to improve parsing, not generation, but, as all the methods involve pre-processing and do not change the grammar itself, we do not affect the generation capabilities of the grammars. We use two of the DELPH-IN wide-coverage grammars: the English Resource Grammar (ERG: Copestake and Flickinger (2000)) and a German grammar, GG (Müller and Kasper, 2000; Crysmann, 2003). We also use the PET parser, and the [incr tsdb()] system profiler and treebanking tool (Oepen, 2001) for evaluation.

### 3 Parser Restriction

An exhaustive parser, such as PET, by default produces every parse licensed by the grammar. However, in many application scenarios, this is unnecessary and time consuming. The benefits of us-

<sup>1</sup><http://wiki.delph-in.net/>

ing a deep parser with a lexicalised grammar are the precision and depth of the analysis produced, but this depth comes from making many fine distinctions which greatly increases the parser search space, making parsing slow. By restricting the lexical items considered during parsing, we improve the efficiency of a parser with a possible trade-off of losing correct parses. For example, the noun phrase reading of *The dog barks* is a correct parse, although unlikely. By blocking the use of *barks* as a noun in this case, we lose this reading. This may be an acceptable trade-off in some applications that can make use of the detailed information, but only if it can be delivered in reasonable time. An example of such an application is the real-time speech translation system developed in the Verbmobil project (Wahlster, 2000), which integrated deep parsing results, where available, into its appointment scheduling and travel planning dialogues. In these experiments we look at two methods of restricting the parser, first by using POS tags and then using lexical types. To control the trade-off between efficiency and precision, we vary which lexical items are restricted according to a likelihood threshold from the respective taggers. Only open class words are restricted, since it is the gross distinctions between, for instance, noun and verb that we would like to utilise. Any differences between categories for closed class words are more subtle and we feel the parser is best left to make these distinctions without restriction. The data set used for these experiments is the *jh5* section of the treebank released with the ERG. This text consists of edited written English in the domain of Norwegian hiking instructions from the LOGON project (Oepen et al., 2004).

#### 3.1 Part of Speech Tags

We use TreeTagger (Schmid, 1994) to produce POS tags and then open class words are restricted if the POS tagger assigned a tag with a probability over a certain threshold. A lower threshold will lead to faster parsing, but at the expense of losing more correct parses. We experiment with various thresholds, and results are shown in Table 1. Since a gold standard treebank for our data set was available, it was possible to evaluate the accuracy of the parser. Evaluation of deep parsing results is often reported only in terms of coverage (number of sentences which re-

Threshold	Coverage	Precision	Time
gold	93.5%	92.2%	N/A
unrestricted	93.3%	92.4%	0.67s
1.00	90.7%	91.9%	0.59s
0.98	88.8%	89.3%	0.49s
0.95	88.4%	89.5%	0.48s
0.90	86.4%	88.5%	0.44s
0.80	84.3%	87.0%	0.43s
0.60	81.5%	87.3%	0.39s

Table 1: Results obtained when restricting the parser lexicon according to the POS tag, where words are restricted according to a threshold of POS probabilities.

ceive an analysis), because, since the hand-crafted grammars are optimised for precision over coverage, the analyses are assumed to be correct. However, in this experiment, we are potentially ‘diluting’ the precision of the grammar by using external resources to remove parses and so it is important that we have some idea of how the accuracy is affected. In the table, precision is the percentage of sentences that, having produced at least one parse, produced a correct parse. A parse was judged to be correct if it exactly matched the gold standard tree in all aspects, syntactic and semantic.

The results show quite clearly how the coverage drops as the average parse time per sentence drops. In hybrid applications that can back-off to less informative analyses, this may be a reasonable trade-off, enabling detailed analyses in shorter times where possible, and using the shallower analyses otherwise.

### 3.2 Lexical Types

Another option for restricting the parser is to use the lexical types used by the grammar itself, in a similar method to that described by Prins and van Noord (2003). This could be considered a form of supertagging as used in LTAG and CCG. Restricting by lexical types should have the effect of reducing ambiguity further than POS tags can do, since one POS tag could still allow the use of multiple lexical items with compatible lexical types. On the other hand, it could be considered more difficult to tag accurately, since there are many more lexical types than POS tags (almost 900 in the ERG) and less training data is available.

Configuration	Coverage	Precision	Time
gold	93.5%	92.2%	N/A
unrestricted	93.3%	92.4%	0.67s
0.98 with POS	93.5%	91.9%	0.63s
0.95 with POS	93.1%	92.4%	0.48s
0.90 with POS	92.9%	92.3%	0.37s
0.80 with POS	91.8%	91.8%	0.31s
0.60 with POS	86.2%	93.5%	0.21s
0.98 no POS	92.9%	92.3%	0.62s
0.95 no POS	90.9%	91.0%	0.48s
0.90 no POS	87.7%	89.2%	0.42s
0.80 no POS	79.7%	84.6%	0.33s
0.60 no POS	67.0%	84.2%	0.23s

Table 2: Results obtained when restricting the parser lexicon according to the predicted lexical type, where words are restricted according to a threshold of tag probabilities. Two models, with and without POS tags as features, were used.

While POS taggers such as TreeTagger are common, and there some supertaggers are available, notably that of Clark and Curran (2007) for CCG, no standard supertagger exists for HPSG. Consequently, we developed a Maximum Entropy model for supertagging using the OpenNLP implementation.<sup>2</sup> Similarly to Zhang and Kordoni (2006), we took training data from the gold-standard lexical types in the treebank associated with ERG (in our case, the July-07 version). For each token, we extracted features in two ways. One used features only from the input string itself: four characters from the beginning and end of the target word token, and two words of context (where available) either side of the target. The second used the features from the first, along with POS tags given by TreeTagger for the context tokens.

We held back the *jh5* section of the treebank for testing the Maximum Entropy model. Again, the lexical items that were to be restricted were controlled by a threshold, in this case the probability given by the maximum entropy model. Table 2 shows the results achieved by these two models, with the unrestricted results and the gold standard provided for comparison.

Here we see the same trends of falling coverage

<sup>2</sup><http://maxent.sourceforge.net/>

with falling time for both models, with the POS tagged model consistently outperforming the word-form model. To give a clearer picture of the comparative performance of all three experiments, Figure 1 shows how the results vary with time for both models, and for the POS tag restricted experiment. Here we can see that the coverage and precision of the lexical type restriction experiment that uses the word-form model is just above that of the POS restricted one. However the POS tagged model clearly outperforms both, showing minimal loss of coverage or precision at a threshold which halved the average parsing time. At the lowest parsing time, we see that precision of the POS tagged model even goes up. This can be explained by noting that coverage here goes down, and obviously we are losing more incorrect parses than correct parses.

This echoes the main result from Prins and van Noord (2003), that filtering the lexical categories used by the parser can significantly reduce parsing time, while maintaining, or even improving, precision. The main differences between our method and that of Prins and van Noord are the training data and the tagging model. The key feature of their experiment was the use of ‘unsupervised’ training data, that is, the uncorrected output of their parser. In this experiment, we used gold standard training data, but much less of it (just under 200 000 words) and still achieved a very good precision. It would be interesting to see what amount of unsupervised parser output we would require to achieve the same level of precision. The other difference was the tagging model, maximum entropy versus Hidden Markov Model (HMM). We selected maximum entropy because Zhang and Kordoni (2006) had shown that they got better results using a maximum entropy tagger instead of a HMM one when predicting lexical types, albeit for a slightly different purpose. It is not possible to directly compare results between our experiments and those in Prins and van Noord, because of different languages, data sets and hardware, but it is worth noting that parsing times are much lower in our setup, perhaps more so than can be attributed to 4 years hardware improvement. While the range of sentence lengths appears to be very similar between the data sets, one possible reason for this could be the very large number of lexical categories used in their ALPINO system.

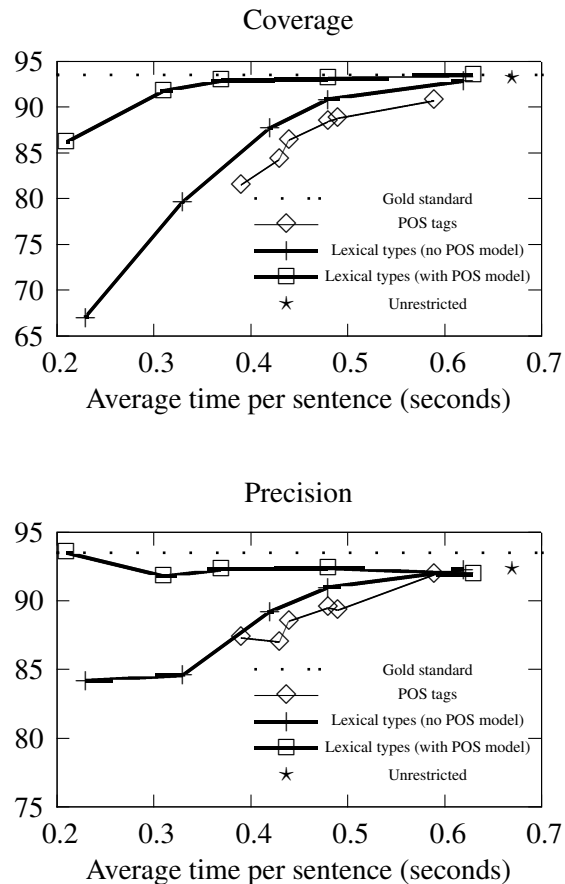


Figure 1: Coverage and precision varying with time for the three restriction experiments. Gold standard and unrestricted results shown for comparison.

While this experiment is similar to that of Clark and Curran (2007), it differs in that their supertagger assign categories to every word, while we look up every word in the lexicon and the tagger is used to filter what the lexicon returns, only if the tagger confidence is sufficiently high. As Table 2 shows, when we use the tags for which the tagger had a low confidence, we lose significant coverage. In order to run as a supertagger rather than a filter, the tagger would need to be much more accurate. While we can look at multi-tagging as an option, we believe much more training data would be needed to achieve a sufficient level of tag accuracy.

Increasing efficiency is important for enabling these heavily lexicalised grammars to bring the benefits of their deep analyses to applications, but simi-

larly important is robustness. The following section is aimed at addressing this issue of robustness, again by using external information.

#### 4 Unknown Word Handling

The lexical information available to the parser is what makes the depth of the analysis possible, and the default configuration of the parser uses an all-or-nothing approach, where a parse is not produced if all the lexical information is not available. However, in order to increase robustness, it is possible to use underspecified lexical information where a fully specified lexical item is not available. One method of doing this, built in to the PET parser, is to use POS tags to select generic lexical items, and hence allow a (less than fully specified) parse to be built.

The six data sets used for these experiments were chosen to give a range of languages and genres. Four sets are English text: *jh5* described in Section 3; *trec* consisting of questions from TREC and included in the treebanks released with the ERG; *a00* which is taken from the BNC and consists of factsheets and newsletters; and *depbank*, the 700 sentences of the Briscoe and Carroll version of DepBank (Briscoe and Carroll, 2006) taken from the Wall Street Journal. The last two data sets are German text: *clef700* consisting of German questions taken from the CLEF competition and *eiche564* a sample of sentences taken from a treebank parsed with the German HPSG grammar, GG and consisting of transcribed German speech data concerning appointment scheduling from the Verbmobil project. Vital statistics of these data sets are described in Table 3.

We used TreeTagger to POS tag the six data sets, with the tagger configured to assign multiple tags, where the probability of the less likely tags was at least half that of the most likely tag. The data was input using a PET input chart (PIC), which allows POS tags to be assigned to each token, and then parsed each with the PET parser.<sup>3</sup> All English data sets used the July-07 CVS version of the ERG and the German sets used the September\_2007 version of GG. Unlike the experiments described in Section 3, adding POS tags in this way will have no effect on sentences which the parser is already able

	Language	Number of Sentences	Ave. Sentence Length
<i>jh5</i>	English	464	14.2
<i>trec</i>	English	693	6.9
<i>a00</i>	English	423	17.2
<i>depbank</i>	English	700	21.5
<i>clef</i>	German	700	7.5
<i>eiche564</i>	German	564	11.5

Table 3: Data sets used in input annotation experiments.

to parse. The POS tags will only be considered when the parser has no lexicon entry for a given word, and hence can only increase coverage. Results are shown in Table 4, comparing the coverage over each set to that obtained without using POS tags to handle unknown words. Coverage here is defined as the percentage of sentences with at least one parse.

These results show very clearly one of the potential drawbacks of using a highly lexicalised grammar formalism like HPSG: unknown words are one of the main causes of parse failure, as quantified in Baldwin et al. (2004) and Nicholson et al. (2008). In the results here, we see that for *jh5*, *trec* and *eiche564*, adding unknown word handling made almost no difference, since the grammars (specifically the lexicons) have been tuned for these data sets. On the other hand, over unseen texts, adding unknown word handling made a dramatic difference to the coverage. This motivates strategies like the POS tag annotation used here, as well as the work on deep lexical acquisition (DLA) described in Zhang and Kordoni (2006) and Baldwin (2005), since no grammar could ever hope to cover all words used within a language.

As mentioned in Section 3, coverage is not the only evaluation metric that should be considered, particularly when adding potentially less precise information to the parsing process (in this case POS tags). Since the primary effect of adding POS tags is shown with those data sets for which we do not have gold standard treebanks, evaluating accuracy in this case is more difficult. However, in order to give some idea of the effects on precision, a sample of 100 sentences from the *a00* data set was evaluated for accuracy, for this and the following experiments.

<sup>3</sup>Subversion revision 384

In this instance, we found there was only a slight drop in precision, where the original analyses had a precision of 82% and the precision of the analyses when POS tags were used was 80%.

Since the parser has the means to accept named entity (NE) information in the input, we also experimented with using generic lexical items generated from NE data. We used SProUT (Becker et al., 2002) to tag the data sets and used PET’s inbuilt NE handling mechanism to add NE items to the input, associated with the appropriate word tokens. This works slightly differently from the POS annotation mechanism, in that NE items are considered by the parser, even when the associated words are in the lexicon. This has the effect of increasing the number of analyses produced for sentences that already have a full lexical span, but could also increase coverage by enabling parses to be produced where there is no lexical span, or where no parse was possible because a token was not recognised as part of a name. In order to isolate the effect of the NE data, we ran one experiment where the input was annotated only with the SProUT data, and another where the POS tags were also added. These results are also in Table 4.

Again, we see coverage increases in the three unseen data sets, *a00*, *depbank* and *clef*, but not to the same extent as the POS tags. Examining the results in more detail, we find that the increases come almost exclusively from sentences without lexical span, rather than in sentences where a token was previously not recognised as part of a name. This means that the NE tagger is operating almost like a POS tagger that only tags proper nouns, and as the POS tagger tags proper nouns quite accurately, we find the NE tagger gives no benefit here. When examining the precision over our sample evaluation set from *a00*, we find that using the NE data alone adds no correct parses, while using NE data with POS tags actually removes correct parses when compared with POS alone, since the (in these cases, incorrect) NE data is preferred over the POS tags. It is possible that another named entity tagger would give better results, and this may be looked at in future experiments.

Other forms of external information might also be used to increase lexical coverage. Zhang and Kordoni (2006) reported a 20% coverage increase over baseline using a lexical type predictor for unknown

words, and so we explored this avenue. The same maximum entropy tagger used in Section 3 was used and each open class word was tagged with its most likely lexical type, as predicted by the maximum entropy model. Table 5 shows the results, with the baseline and POS annotated results for comparison. As with the previous experiments, we see a coverage increase in those data sets which are considered unseen text for these grammars. Again it is clear that the use of POS tags as features obviously improves the maximum entropy model, since this second model has almost 10% better coverage on our unseen texts. However, lexical types do not appear to be as effective for increasing lexical coverage as the POS tags. One difference between the POS and lexical type taggers is that the POS tagger could produce multiple tags per word. Therefore, for the next experiment, we altered the lexical type tagger so it could also produce multiple tags. As with the Tree-Tagger configuration we used for POS annotation, extra lexical type tags were produced if they were at least half as probable as the most likely tag. A lower probability threshold of 0.01 was set, so that hundreds of tags of equal likelihood were not produced in the case where the tagger was unable to make an informed prediction. The results with multiple tagging are also shown in Table 5.

The multiple tagging version gives a coverage increase of between 2 and 10% over the single tag version of the tagger, but, at least for the English data sets, it is still less effective than straight-forward POS tagging. For the German unseen data set, *clef*, we do start getting above what the POS tagger can achieve. This may be in part because of the features used by the lexical type tagger — German, being a more morphologically rich language, may benefit more from the prefix and suffix features used in the tagger.

In terms of precision measured on our sample evaluation set, the single tag version of the lexical type tagger which used POS tag features achieved a very good precision of 87% where, of all the extra sentences that could now be parsed, only one did not have a correct parse. In an application where precision is considered much more important than coverage, this would be a good method of increasing coverage without loss of accuracy. The single tag version that did not use POS tags in the model achieved

	Baseline	with POS	NE only	NE+POS
jh5	93.1%	93.3%	93.1%	93.3%
trec	97.1%	97.5%	97.4%	97.7%
a00	50.1%	83.9%	53.0%	85.8%
depbank	36.3%	76.9%	51.1%	80.4%
clef	22.0%	67.7%	42.3%	75.3%
eiche564	63.8%	63.8%	64.0%	64.0%

Table 4: Parser coverage with baseline using no unknown word handling and unknown word handling using POS tags, SProUT named entity data as the only annotation, or SProUT tags in addition to POS annotation.

	Baseline	Single Lexical Types			Multiple Lexical Types	
		POS	-POS	+POS	-POS	+POS
jh5	93.1%	93.3%	93.3%	93.3%	93.5%	93.5%
trec	97.1%	97.5%	97.3%	97.4%	97.3%	97.4%
a00	50.1%	83.9%	63.8%	72.6%	65.7%	78.5%
depbank	36.3%	76.9%	51.7%	64.4%	53.9%	69.7%
clef	22.0%	67.7%	59.9%	66.8%	69.7%	76.9%
eiche564	63.8%	63.8%	63.8%	63.8%	63.8%	63.8%

Table 5: Parser coverage using a lexical type predictor for unknown word handling. The predictor was run in single tag mode, and then in multi-tag mode. Two different tagging models were used, with and without POS tags as features.

the same precision as with using only POS tags, but without the same increase in coverage. On the other hand, the multiple tagging versions, which at least started approaching the coverage of the POS tag experiment, dropped to a precision of around 76%.

From the results of Section 3, one might expect that at least the lexical type method of handling unknown words might at least lead to quicker parsing than when using POS tags, however POS tags are used differently in this situation. When POS tags are used to restrict the parser, any lexicon entry that unifies with the generic part-of-speech lexical category can be used by the parser. That is, when the word is restricted to, for example, a *verb*, any lexical item with one of the numerous more specific verb categories can be used. In contrast, in these experiments, the lexicon plays no part. The POS tag causes one underspecified lexical item (per POS tag) to be considered in parsing. While these underspecified items may allow more analyses to be built than if the exact category was used, the main contribution to parsing time turned out to be the number of tags assigned to each word, whether that was a POS tag or a lexical type. The POS tagger assigned multiple tags much less frequently than the multiple tagging

lexical type tagger and so had a faster average parsing time. The single tagging lexical type tagger had only slightly fewer tags assigned overall, and hence was slightly faster, but at the expense of a significantly lower coverage.

## 5 Conclusion

The work reported here shows the benefits that can be gained by utilising external resources to annotate parser input in highly lexicalised grammar formalisms. Even something as simple and readily available (for languages likely to have lexicalised grammars) as a POS tagger can massively increase the parser coverage on unseen text. While annotating with named entity data or a lexical type supertagger were also found to increase coverage, the POS tagger had the greatest effect with up to 45% coverage increase on unseen text.

In terms of efficiency, POS tags were also shown to speed up parsing by filtering unlikely lexicon items, but better results were achieved in this case by using a lexical type supertagger. Again encouraging the use of external resources, the supertagging was found to be much more effective when POS tags

were used to train the tagging model, and in this configuration, managed to halve the parsing time with minimal effect on coverage or precision.

## 6 Further Work

A number of avenues of future research were suggested by the observations made during this work. In terms of robustness and increasing lexical coverage, more work into using lexical types for unknown words could be explored. In light of the encouraging results for German, one area to look at is the effect of different features for different languages. Use of back-off models might also be worth considering when the tagger probabilities are low.

Different methods of using the supertagger could also be explored. The experiment reported here used the single most probable type for restricting the lexicon entries used by the parser. Two extensions of this are obvious. The first is to use multiple tags over a certain threshold, by either inputting multiple types as was done for the unknown word handling, or by using a generic type that is compatible with all the predicted types over a certain threshold. The other possible direction to try is to not check the predicted type against the lexicon, but to simply construct a lexical item from the most likely type, given a (high) threshold probability. This would be similar to the CCG supertagging mechanism and is likely to give generous speedups at the possible expense of precision, but it would be illuminating to discover how this trade-off plays out in our setup.

## References

- Timothy Baldwin, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Open. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2047–50, Lisbon, Portugal.
- Timothy Baldwin. 2005. Bootstrapping deep lexical resources: Resources for courses. In *Proceedings of the ACL-SIGLEX 2005 Workshop on Deep Lexical Acquisition*, pages 67–76, Ann Arbor, USA.
- Srinivas Bangalore and Aravind K. Joshi. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the 15th COLING Conference*, pages 154–160, Kyoto, Japan.
- Markus Becker, Witold Drozdowski, 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.
- Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalised statistical parser on the PARC DepBank. In *Proceedings of the 44th Annual Meeting of the ACL*, pages 41–48, Sydney, Australia.
- Ulrich Callmeier. 2000. PET - a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–107.
- Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.
- Stephen Clark. 2002. Supertagging for combinatory categorical grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammar and Related Frameworks*, pages 101–106, Venice, Italy.
- Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.
- Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Annual Meeting of the ACL and 10th Conference of the EACL (ACL-EACL 2001)*, Toulouse, France.
- Berthold Crysmann. 2003. On the efficient implementation of German verb placement in HPSG. In *Proceedings of RANLP 2003*, pages 112–116, Borovets, Bulgaria.
- Julia Hockenmaier, Gann Bierner, and Jason Baldridge. 2002. Extending the coverage of a CCG system. *Research in Language and Computation*.
- Bernd Kiefer, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. 1999. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 473–480, Maryland, USA.
- Stefan Müller and Walter Kasper. 2000. HPSG analysis of German. In *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 238–253. Springer, Berlin, Germany.
- Jeremy Nicholson, Valia Kordoni, Yi Zhang, Timothy Baldwin, and Rebecca Dridan. 2008. Evaluating and extending the coverage of HPSG grammars. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco.



- Ruth O'Donovan, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. 2005. Large-scale induction and evaluation of lexical resources from the Penn-II and Penn-III treebanks. *Computational Linguistics*, 31:pp 329–366.
- Stephan Oepen, Helge Dyvik, Jan Tore Lønning, Erik Velldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, and Victoria Rosén. 2004. Somå kapp-ete med trollet? Towards MRS-based Norwegian—English machine translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, Baltimore, USA.
- Stephan Oepen. 2001. [incr tsdb()] – competence and performance laboratory. User manual, Computational Linguistics, Saarland University, Saarbrücken, Germany.
- Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, USA.
- Robbert Prins and Gertjan van Noord. 2003. Reinforcing parser preferences through tagging. *Traitement Automatique des Langues*, 44(3):121–139.
- Kenji Sagae, Yusuke Miyao, and Jun'ichi Tsujii. 2007. HPSG parsing with shallow dependency constraints. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 624–631, Prague, Czech Republic.
- Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.
- Gertjan van Noord and Robert Malouf. 2004. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop Beyond Shallow Analyses – Formalisms and statistical modelling for deep analyses*.
- Wolfgang Wahlster, editor. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer-Verlag, Berlin.
- Yi Zhang and Valia Kordoni. 2006. Automated deep lexical acquisition for robust open texts processing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 275–280, Genoa, Italy.

# Assessing Dialog System User Simulation Evaluation Measures Using Human Judges

**Hua Ai**

University of Pittsburgh  
Pittsburgh PA, 15260, USA  
hua@cs.pitt.edu

**Diane J. Litman**

University of Pittsburgh  
Pittsburgh, PA 15260, USA  
litman@cs.pitt.edu

## Abstract

Previous studies evaluate simulated dialog corpora using evaluation measures which can be automatically extracted from the dialog systems' logs. However, the validity of these automatic measures has not been fully proven. In this study, we first recruit human judges to assess the quality of three simulated dialog corpora and then use human judgments as the gold standard to validate the conclusions drawn from the automatic measures. We observe that it is hard for the human judges to reach good agreement when asked to rate the quality of the dialogs from given perspectives. However, the human ratings give consistent ranking of the quality of simulated corpora generated by different simulation models. When building prediction models of human judgments using previously proposed automatic measures, we find that we cannot reliably predict human ratings using a regression model, but we can predict human rankings by a ranking model.

## 1 Introduction

User simulation has been widely used in different phases in spoken dialog system development. In the system development phase, user simulation is used in training different system components. For example, (Levin et al., 2000) and (Scheffler, 2002) exploit user simulations to generate large corpora for using Reinforcement Learning to develop dialog strategies, while (Chung, 2004) implement user simulation to train the speech recognizer and understanding components.

While user simulation is considered to be more low-cost and time-efficient than experiments with human subjects, one major concern is how well the state-of-the-art user simulations can mimic human user behaviors and how well they can substitute for human users in a variety of tasks. (Schatzmann et al., 2005) propose a set of evaluation measures to assess the quality of simulated corpora. They find that these evaluation measures are sufficient to discern simulated from real dialogs. Since this multiple-measure approach does not offer a easily reportable statistic indicating the quality of a user simulation, (Williams, 2007) proposes a single measure for evaluating and rank-ordering user simulations based on the divergence between the simulated and real users' performance. This new approach also offers a lookup table that helps to judge whether an observed ordering of two user simulations is statistically significant.

In this study, we also strive to develop a prediction model of the rankings of the simulated users' performance. However, our approach use human judgments as the gold standard. Although to date there are few studies that use human judges to directly assess the quality of user simulation, we believe that this is a reliable approach to assess the simulated corpora as well as an important step towards developing a comprehensive set of user simulation evaluation measures. First, we can estimate the difficulty of the task of distinguishing real and simulated corpora by knowing how hard it is for human judges to reach an agreement. Second, human judgments can be used as the gold standard of the automatic evaluation measures. Third, we can validate the automatic

measures by correlating the conclusions drawn from the automatic measures with the human judgments.

In this study, we recruit human judges to assess the quality of three user simulation models. Judges are asked to read the transcripts of the dialogs between a computer tutoring system and the simulation models and to rate the dialogs on a 5-point scale from different perspectives. Judges are also given the transcripts between human users and the computer tutor. We first assess human judges' abilities in distinguishing real from simulated users. We find that it is hard for human judges to reach good agreement on the ratings. However, these ratings give consistent ranking on the quality of the real and the simulated user models. Similarly, when we use previously proposed automatic measures to predict human judgments, we cannot reliably predict human ratings using a regression model, but we can consistently mimic human judges' rankings using a ranking model. We suggest that this ranking model can be used to quickly assess the quality of a new simulation model without manual efforts by ranking the new model against the old models.

## 2 Related Work

A lot of research has been done in evaluating different components of Spoken Dialog Systems as well as overall system performance. Different evaluation approaches are proposed for different tasks. Some studies (e.g., (Walker et al., 1997)) build regression models to predict user satisfaction scores from the system log as well as the user survey. There are also studies that evaluate different systems/system components by ranking the quality of their outputs. For example, (Walker et al., 2001) train a ranking model that ranks the outputs of different language generation strategies based on human judges' rankings. In this study, we build both a regression model and a ranking model to evaluate user simulation.

(Schatzmann et al., 2005) summarize some broadly used automatic evaluation measures for user simulation and integrate several new automatic measures to form a comprehensive set of statistical evaluation measures. The first group of measures investigates how much information is transmitted in the dialog and how active the dialog participants are. The second group of measures analyzes the style of

the dialog and the last group of measures examines the efficiency of the dialogs. While these automatic measures are handy to use, these measures have not been validated by humans.

There are well-known practices which validate automatic measures using human judgments. For example, in machine translation, BLEU score (Papineni et al., 2002) is developed to assess the quality of machine translated sentences. Statistical analysis is used to validate this score by showing that BLEU score is highly correlated with the human judgment. In this study, we validate a subset of the automatic measures proposed by (Schatzmann et al., 2005) by correlating the measures with human judgments. We follow the design of (Linguistic Data Consortium, 2005) in obtaining human judgments. We call our study an assessment study.

## 3 System and User Simulation Models

In this section, we describe our dialog system (IT-SPOKE) and the user simulation models which we use in the assessment study. IT-SPOKE is a speech-enabled Intelligent Tutoring System that helps students understand qualitative physics questions. In the system, the computer tutor first presents a physics question and the student types an essay as the answer. Then, the tutor analyzes the essay and initiates a tutoring dialog to correct misconceptions and to elicit further explanations. A corpus of 100 tutoring dialogs was collected between 20 college students (solving 5 physics problems each) and the computer tutor, yielding 1388 student turns. The correctness of student answers is automatically judged by the system and kept in the system's logs. Our previous study manually clustered tutor questions into 20 clusters based on the knowledge (e.g., acceleration, Newton's 3rd Law) that is required to answer each question (Ai and Litman, 2007).

We train three simulation models from the real corpus: the random model, the correctness model, and the cluster model. All simulation models generate student utterances on the word level by picking out the recognized student answers (with potential speech recognition errors) from the human subject experiments with different policies. The **random model (ran)** is a simple unigram model which randomly picks a student's utterance from the real cor-

pus as the answer to a tutor’s question, neglecting which question it is. The **correctness model (cor)** is designed to give a correct/incorrect answer with the same probability as the average of real students. For each tutor’s question, we automatically compute the average correctness rate of real student answers from the system logs. Then, a correct/incorrect answer is randomly chosen from the correct/incorrect answer sets for this question. The **cluster model (clu)** tries to model student learning by assuming that a student will have a higher chance to give a correct answer to the question of a cluster in which he/she mostly answers correctly before. It computes the conditional probability of whether a student answer is correct/incorrect given the content of the tutor’s question and the correctness of the student’s answer to the last previous question that belongs to the same question cluster. We also refer to the real student as the **real student model (real)** in the paper. We hypothesize that the ranking of the four student models (from the most realistic to the least) is: *real*, *clu*, *cor*, and *ran*.

## 4 Assessment Study Design

### 4.1 Data

We decided to conduct a middle-scale assessment study that involved 30 human judges. We conducted a small pilot study to estimate how long it took a judge to answer all survey questions (described in Section 4.2) in one dialog because we wanted to control the length of the study so that judges would not have too much cognitive load and would be consistent and accurate on their answers. Based on the pilot study, we decided to assign each judge 12 dialogs which took about an hour to complete. Each dialog was assigned to two judges. We used three out of the five physics problems from the original real corpus to ensure the variety of dialog contents while keeping the corpus size small. Therefore, the evaluation corpus consisted of 180 dialogs, in which 15 dialogs were generated by each of the 4 student models on each of the 3 problems.

## 4.2 Survey Design

### 4.2.1 Survey questions

We designed a web survey to collect human judgments on a 5-point scale on both utterance and di-

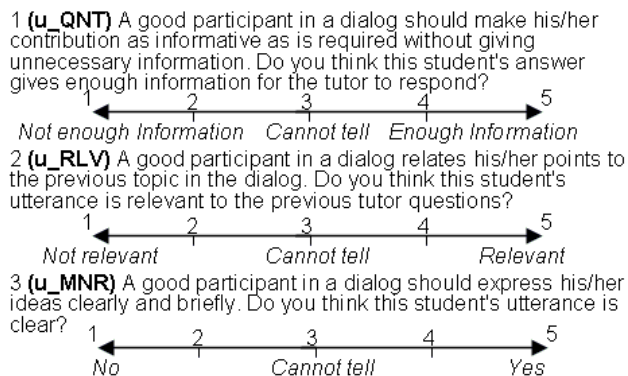


Figure 1: Utterance level questions.

alog levels. Each dialog is separated into pairs of a tutor question and the corresponding student answer. Figure 1 shows the three questions which are asked for each tutor-student utterance pair. The three questions assess the quality of the student answers from three aspects of Grice’s Maxim (Grice, 1975): Maxim of Quantity (**u\_QNT**), Maxim of Relevance (**u\_RLV**), and Maxim of Manner (**u\_MNR**). We do not include the Maxim of Quality because in our task domain the correctness of the student answers depends largely on students’ physics knowledge, which is not a factor we would like to consider when evaluating the realness of the students’ dialog behaviors.

In Figure 2, we show the three dialog level questions which are asked at the end of each dialog. The first question (**d\_TUR**) is a Turing test type of question which aims to obtain an impression of the student’s overall performance. The second question (**d\_QLT**) assesses the dialog quality from a tutoring perspective. The third question (**d\_PAT**) sets a higher standard on the student’s performance. Unlike the first two questions which ask whether the student “looks” good, this question further asks whether the judges would like to partner with the particular student.

### 4.2.2 Survey Website

We display one tutor-student utterance pair and the three utterance level questions on each web page. After the judges answer the three questions, he/she will be led to the next page which displays the next pair of tutor-student utterances in the dialog with the same three utterance level questions. The judge

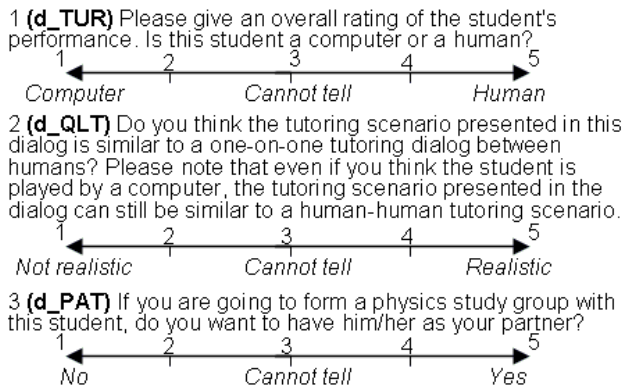


Figure 2: Dialog level questions.

reads through the dialog in this manner and answers all utterance level questions. At the end of the dialog, three dialog level questions are displayed on one webpage. We provide a textbox under each dialog level question for the judge to type in a brief explanation on his/her answer. After the judge completes the three dialog level questions, he/she will be led to a new dialog. This procedure repeats until the judge completes all of the 12 assigned dialogs.

### 4.3 Assessment Study

30 college students are recruited as human judges via flyers. Judges are required to be native speakers of American English to make correct judgments on the language use and fluency of the dialog. They are also required to have taken at least one course on Newtonian physics to ensure that they can understand the physics tutoring dialogs and make judgments about the content of the dialogs. We follow the same task assigning procedure that is used in (Linguistic Data Consortium, 2005) to ensure a uniform distribution of judges across student models and dialogs while maintaining a random choice of judges, models, and dialogs. Judges are instructed to work as quickly as comfortably possible. They are encouraged to provide their intuitive reactions and not to ponder their decisions.

## 5 Assessment Study Results

In the initial analysis, we observe that it is a difficult task for human judges to rate on the 5-point scale and the agreements among the judges are fairly low. Table 1 shows for each question, the percentages of

d_TUR	d_QLT	d_PAT	u_QNT	u_RLV	u_MNR
22.8%	27.8%	35.6%	39.2%	38.4%	38.7%

Table 1: Percent agreements on 5-point scale

pairs of judges who gave the same ratings on the 5-point scale. For the rest of the paper, we collapse the “definitely” types of answers with its adjacent “probably” types of answers (more specifically, answer 1 with 2, and 4 with 5). We substitute scores 1 and 2 with a score of 1.5, and scores 4 and 5 with a score of 4.5. A score of 3 remains the same.

### 5.1 Inter-annotator agreement

Table 2 shows the inter-annotator agreements on the collapsed 3-point scale. The first column presents the question types. In the first row, “diff” stands for the differences between human judges’ ratings. The column “diff=0” shows the percent agreements on the 3-point scale. We can see the improvements from the original 5-point scale when comparing with Table 1. The column “diff=1” shows the percentages of pairs of judges who agree with each other on a weaker basis in that one of the judges chooses “cannot tell”. The column “diff=2” shows the percentages of pairs of judges who disagree with each other. The column “Kappa” shows the un-weighted kappa agreements and the column “Kappa\*” shows the linear weighted kappa. We construct the confusion matrix for each question to compute kappa agreements. Table 3 shows the confusion matrix for d\_TUR. The first three rows of the first three columns show the counts of judges’ ratings on the 3-point scale. For example, the first cell shows that there are 20 cases where both judges give 1.5 to the same dialog. When calculating the linear weighted kappa, we define the distances between the adjacent categories to be one<sup>1</sup>. Note that we randomly picked two judges to rate each dialog so that different dialogs are rated by different pairs of judges and one pair of judges only worked on one dialog together. Thus, the kappa agreements here do not reflect the agreement of one pair of judges. Instead, the kappa agreements show the overall observed agreement among every pair of

<sup>1</sup>We also calculated the quadratic weighted kappa in which the distances are squared and the kappa results are similar to the linear weighted ones. For calculating the two weighted kappas, see <http://faculty.vassar.edu/lowry/kappa.html> for details.

Q	diff=0	diff=1	diff=2	Kappa	Kappa*
d_TUR	35.0%	45.6%	19.4%	0.022	0.079
d_QLT	46.1%	28.9%	25.0%	0.115	0.162
d_PAT	47.2%	30.6%	22.2%	0.155	0.207
u_QNT	66.8%	13.9%	19.3%	0.377	0.430
u_RLV	66.6%	17.2%	16.2%	0.369	0.433
u_MNR	67.5%	15.4%	17.1%	0.405	0.470

Table 2: Agreements on 3-point scale

	score=1.5	score=3	score=4.5	sum
score=1.5	20	26	20	66
score=3	17	11	19	47
score=4.5	15	20	32	67
sum	52	57	71	180

Table 3: Confusion Matrix on d\_TUR

judges controlling for the chance agreement.

We observe that human judges have low agreement on all types of questions, although the agreements on the utterance level questions are better than the dialog level questions. This observation indicates that assessing the overall quality of simulated/real dialogs on the dialog level is a difficult task. The lowest agreement appears on d\_TUR. We investigate the low agreements by looking into judges' explanations on the dialog level questions. 21% of the judges find it hard to rate a particular dialog because that dialog is too short or the student utterances mostly consist of one or two words. There are also some common false beliefs among the judges. For example, 16% of the judges think that humans will say longer utterances while 9% of the judges think that only humans will admit the ignorance of an answer.

## 5.2 Rankings of the models

In Table 4, the first column shows the name of the questions; the second column shows the name of the models; the third to the fifth column present the percentages of judges who choose answer 1 and 2, can't tell, and answer 4 and 5. For example, when looking at the column "1 and 2" for d\_TUR, we see that 22.2% of the judges think a dialog by a real student is generated probably or definitely by a computer; more judges (25.6%) think a dialog by the cluster model is generated by a computer; even more judges (32.2%) think a dialog by the correctness model is generated by a computer; and even

Question	model	1 and 2	can't tell	4 and 5
d_TUR	real	22.2%	28.9%	48.9%
	clu	25.6%	31.1%	43.3%
	cor	32.2%	26.7%	41.1%
	ran	51.1%	28.9%	20.0%
d_QLT	real	20.0%	10.0%	70.0%
	clu	21.1%	20.0%	58.9%
	cor	24.4%	15.6%	60.0%
	ran	60.0%	18.9%	21.1%
d_PAT	real	28.9%	21.1%	50.0%
	clu	41.1%	17.8%	41.1%
	cor	43.3%	18.9%	37.8%
	ran	82.2%	14.4%	3.4%

Table 4: Rankings on Dialog Level Questions

more judges (51.1%) think a dialog by the random model is generated by a computer. When looking at the column "4 and 5" for d\_TUR, we find that most of the judges think a dialog by the real student is generated by a human while the fewest number of judges think a dialog by the random model is generated by a human. Given that more human-like is better, both rankings support our hypothesis that the quality of the models from the best to the worst is: *real*, *clu*, *cor*, and *ran*. In other words, although it is hard to obtain well-agreed ratings among judges, we can combine the judges' ratings to produce the ranking of the models. We see consistent ranking orders on d\_QLT and d\_PAT as well, except for a disorder of cluster and correctness model on d\_QLT indicated by the underlines.

When comparing two models, we can tell which model is better from the above rankings. Nevertheless, we also want to know how significant the difference is. We use t-tests to examine the significance of differences between every two models. We average the two human judges' ratings to get an averaged score for each dialog. For each pair of models, we compare the two groups of the averaged scores for the dialogs generated by the two models using 2-tail t-tests at the significance level of  $p < 0.05$ . In Table 5, the first row presents the names of the models in each pair of comparison. **Sig** means that the t-test is significant after Bonferroni correction; question mark (?) means that the t-test is significant before the correction, but not significant afterwards, we treat this situation as a trend; **not** means that the t-test is not significant at all. The table shows

	real-ran	real-cor	real-clu	ran-cor	ran-clu	cor-clu
d_TUR	sig	not	not	sig	sig	not
d_QLT	sig	not	not	sig	sig	not
d_PAT	sig	?	?	sig	sig	not
u_QNT	sig	not	not	sig	sig	not
u_RLV	sig	not	not	sig	sig	not
u_MNR	sig	not	not	sig	sig	not

Table 5: T-Tests Results

that only the random model is significantly different from all other models. The correctness model and the cluster model are not significantly different from the real student given the human judges’ ratings, neither are the two models significantly different from each other.

### 5.3 Human judgment accuracy on d\_TUR

We look further into d\_TUR in Table 4 because it is the only question that we know the ground truth. We compute the accuracy of human judgment as (number of ratings 4&5 on real dialogs + number of ratings of 1&2 on simulated dialogs)/(2\*total number of dialogs). The accuracy is 39.44%, which serves as further evidence that it is difficult to discern human from simulated users directly. A weaker accuracy is calculated to be 68.35% when we treat “cannot tell” as a correct answer as well.

## 6 Validating Automatic Measures

Since it is expensive to use human judges to rate simulated dialogs, we are interested in building prediction models of human judgments using automatic measures. If the prediction model can reliably mimic human judgments, it can be used to rate new simulation models without collecting human ratings. In this section, we use a subset of the automatic measures proposed in (Schatzmann et al., 2005) that are applicable to our data to predict human judgments. Here, the human judgment on each dialog is calculated as the average of the two judges’ ratings. We focus on predicting human judgments on the dialog level because these ratings represent the overall performance of the student models. We use six high-level dialog feature measures including the number of student turns (**Sturn**), the number of tutor turns (**Tturn**), the number of words per stu-

dent turn (**Swordrate**), the number of words per tutor turn (**Twordrate**), the ratio of system/user words per dialog (**WordRatio**), and the percentage of correct answers (**cRate**).

### 6.1 The Regression Model

We use stepwise multiple linear regression to model the human judgments using the set of automatic features we listed above. The stepwise procedure automatically selects measures to be included in the model. For example, d\_TUR is predicted as  $3.65 - 0.08 * WordRatio - 3.21 * Swordrate$ , with an R-square of 0.12. The prediction models for d\_QLT and d\_PAT have similar low R-square values of 0.08 and 0.17, respectively. This result is not surprising because we only include the surface level automatic measures here. Also, these measures are designed for comparison between models instead of prediction. Thus, in Section 6.2, we build a ranking model to utilize the measures in their comparative manner.

### 6.2 The Ranking Model

We train three ranking models to mimic human judges’ rankings of the real and the simulated student models on the three dialog level questions using RankBoost, a boosting algorithm for ranking ((Freund et al., 2003), (Mairesse et al., 2007)). We briefly explain the algorithm using the same terminologies and equations as in (Mairesse et al., 2007), by building the ranking model for d\_TUR as an example. In the training phase, the algorithm takes as input a group of dialogs that are represented by values of the automatic measures and the human judges’ ratings on d\_TUR. The RankBoost algorithm treats the group of dialogs as ordered pairs:

$$\mathcal{T} = \{(x, y) \mid \begin{array}{l} x, y \text{ are two dialog samples,} \\ x \text{ has a higher human rated score than } y \end{array}\}$$

Each dialog  $x$  is represented by a set of  $m$  indicator functions  $h_s(x)$  ( $1 \leq s \leq m$ ). For example:

$$h_s(x) = \begin{cases} 1 & \text{if WordRatio}(x) \geq 0.47 \\ 0 & \text{otherwise} \end{cases}$$

Here, the threshold of 0.47 is calculated by RankBoost.  $\alpha$  is a parameter associated with each indicator function. For each dialog, a ranking score is

calculated as:

$$F(x) = \sum_s \alpha_s h_s(x) \quad (1)$$

In the training phase, the human ratings are used to set  $\alpha$  by minimizing the loss function:

$$LOSS = \frac{1}{|T|} \sum_{(x,y) \in T} eval(F(x) \leq F(y)) \quad (2)$$

The *eval* function returns 0 if  $(x, y)$  pair is ranked correctly, and 1 otherwise. In other words, **LOSS** score is the percentage of misordered pairs where the order of the predicted scores disagree with the order indicated by human judges. In the testing phase, the ranking score for every dialog is calculated by Equation 1. A baseline model which ranks dialog pairs randomly produces a LOSS of 0.5 (lower is better).

While LOSS indicates how many pairs of dialogs are ranked correctly, our main focus here is to rank the performance of the four student models instead of individual dialogs. Therefore, we propose another Averaged Model Ranking (**AMR**) score. AMR is computed as the sum of the ratings of all the dialogs generated by one model averaged by the number of the dialogs. The four student models are then ranked based on their AMR scores. The chance to get the right ranking order of the four student models by random guess is  $1/(4!)$ .

Table 6 shows a made-up example to illustrate the two measures. *real\_1* and *real\_2* are two dialogs generated by the real student model; *ran\_1* and *ran\_2* are two dialogs by the random model. The second and third column shows the human-rated score as the gold standard and the machine-predicted score in the testing phase respectively. The LOSS in this example is  $1/6$ , because only the pair of *real\_2* and *ran\_1* is misordered out of all the 6 possible pair combinations. We then compute the AMR of the two models. According to human-rated scores, the real model is scored  $0.75 (= (0.9+0.6)/2)$  while the random model is scored  $0.3$ . When looking at the predicted scores, the real model is scored  $0.65$ , which is also higher than the random model with a score of  $0.4$ . We thus conclude that the ranking model ranks the two student models correctly according to the overall rating measure. We use both LOSS and AMR to evaluate the ranking models.

Dialog	Human-rated Score	Predicted Score
real_1	0.9	0.9
real_2	0.6	0.4
ran_1	0.4	0.6
ran_2	0.2	0.2

Table 6: A Made-up Example of the Ranking Model

Cross Validation	d_TUR	d_QLT	d_PAT
Regular	0.176	0.155	0.151
Minus-one-model	0.224	0.180	0.178

Table 7: LOSS scores for Regular and Minus-one-model (during training) Cross Validations

First, we use regular 4-fold cross validation where we randomly hold out 25% of the data for testing and train on the remaining 75% of the data for 4 rounds. Both the training and the testing data consist of dialogs equally distributed among the four student models. However, since the practical usage of the ranking model is to rank a new model against several old models without collecting additional human ratings, we further test the algorithm by repeating the 4 rounds of testing while taking turns to hold out the dialogs from one model in the training data, assuming that model is the new model that we do not have human ratings to train on. The testing corpus still consists of dialogs from all four models. We call this approach the minus-one-model cross validation.

Table 7 shows the LOSS scores for both cross validations. Using 2-tailed t-tests, we observe that the ranking models significantly outperforms the random baseline in all cases after Bonferroni correction ( $p < 0.05$ ). When comparing the two cross validation results for the same question, we see more LOSS in the more difficult minus-one-model case. However, the LOSS scores do not offer a direct conclusion on whether the ranking model ranks the four student models correctly or not. To address this question, we use AMR scores to re-evaluate all cross validation results. Table 8 shows the human-rated and predicted AMR scores averaged over four rounds of testing on the regular cross validation results. We see that the ranking model gives the same rankings of the student models as the human judges on all questions. When applying AMR on the minus-one-model cross validation results, we see similar results that the ranking model reproduces hu-



	real		clu		cor		ran	
	human	predicted	human	predicted	human	predicted	human	predicted
d_TUR	0.68	0.62	0.65	0.59	0.63	0.52	0.51	0.49
d_QLT	0.75	0.71	0.71	0.63	0.69	0.61	0.48	0.50
d_PAR	0.66	0.65	0.60	0.60	0.58	0.57	0.31	0.32

Table 8: AMR Scores for Regular Cross Validation

man judges' rankings. Therefore, we suggest that the ranking model can be used to evaluate a new simulation model by ranking it against several old models. Since our testing corpus is relatively small, we would like to confirm this result on a large corpus and on other dialog systems in the future.

## 7 Conclusion and Future Work

Automatic evaluation measures are used in evaluating simulated dialog corpora. In this study, we investigate a set of previously proposed automatic measures by comparing the conclusions drawn by these measures with human judgments. These measures are considered as valid if the conclusions drawn by these measures agree with human judgments. We use a tutoring dialog corpus with real students, and three simulated dialog corpora generated by three different simulation models trained from the real corpus. Human judges are recruited to read the dialog transcripts and rate the dialogs by answering different utterance and dialog level questions. We observe low agreements among human judges' ratings. However, the overall human ratings give consistent rankings on the quality of the real and simulated user models. Therefore, we build a ranking model which successfully mimics human judgments using previously proposed automatic measures. We suggest that the ranking model can be used to rank new simulation models against the old models in order to assess the quality of the new model.

In the future, we would like to test the ranking model on larger dialog corpora generated by more simulation models. We would also want to include more automatic measures that may be available in the richer corpora to improve the ranking and the regression models.

## Acknowledgments

This work is supported by NSF 0325054. We thank J. Tereault, M. Rotaru, K. Forbes-Riley and the

anonymous reviewers for their insightful suggestions, F. Mairesse for helping with RankBoost, and S. Silliman for his help in the survey experiment.

## References

- H. Ai and D. Litman. 2007. *Knowledge Consistent User Simulations for Dialog Systems*. In Proc. of Inter-speech 2007.
- G. Chung. 2004. *Developing a Flexible Spoken Dialog System Using Simulation*. In Proc. of ACL 04.
- Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. 2003. *An Efficient Boosting Algorithm for Combining Preferences*. Journal of Machine Learning Research.
- H. P. Grice 1975. *Logic and Conversation*. Syntax and Semantics III: Speech Acts, 41-58.
- E. Levin, R. Pieraccini, and W. Eckert. 2000. *A Stochastic Model of Human-Machine Interaction For learning Dialog Strategies*. IEEE Trans. On Speech and Audio Processing, 8(1):11-23.
- Linguistic Data Consortium. 2005. *Linguistic Data Annotation Specification: Assessment of Fluency and Adequacy in Translations*.
- F. Mairesse, M. Walker, M. Mehl and R. Moore. 2007. *Using Linguistic Cues for the Automatic Recognition of Personality in Conversation and Text*. Journal of Artificial Intelligence Research, Vol 30, pp 457-501.
- K.A. Papineni, S. Roukos, R.T. Ward, and W-J. Zhu. 2002. *Bleu: A Method for Automatic Evaluation of Machine Translation*. In Proc. of 40th ACL.
- J. Schatzmann, K. Georgila, and S. Young. 2005. *Quantitative Evaluation of User Simulation Techniques for Spoken Dialog Systems*. In Proc. of 6th SIGdial.
- K. Scheffler. 2002. *Automatic Design of Spoken Dialog Systems*. Ph.D. diss., Cambridge University.
- J. D. Williams. 2007. *A Method for Evaluating and Comparing User Simulations: The Cramer-von Mises Divergence*. Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU).
- M. Walker, D. Litman, C. Kamm, and A. Abella. 1997. *PARADISE: A Framework for Evaluating Spoken Dialog Agents*. In Proc. of ACL 97.
- M. Walker, O. Rambow, and M. Rogati. 2001. *SPoT: A Trainable Sentence Planner*. In Proc. of NAACL 01.

# Robust Dialog Management with N-best Hypotheses Using Dialog Examples and Agenda

Cheongjae Lee, Sangkeun Jung and Gary Geunbae Lee

Pohang University of Science and Technology  
Department of Computer Science and Engineering  
Pohang, Republic of Korea  
{lcj80, hugman, gblee}@postech.ac.kr

## Abstract

This work presents an agenda-based approach to improve the robustness of the dialog manager by using dialog examples and n-best recognition hypotheses. This approach supports n-best hypotheses in the dialog manager and keeps track of the dialog state using a discourse interpretation algorithm with the agenda graph and focus stack. Given the agenda graph and n-best hypotheses, the system can predict the next system actions to maximize multi-level score functions. To evaluate the proposed method, a spoken dialog system for a building guidance robot was developed. Preliminary evaluation shows this approach would be effective to improve the robustness of example-based dialog modeling.

## 1 Introduction

Development of spoken dialog systems involves human language technologies which must cooperate to answer user queries. Since the performance in human language technologies such as Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU)<sup>1</sup> have been improved, this advance has made it possible to develop spoken dialog systems for many different application domains.

Nevertheless, there are major problems for practical spoken dialog systems. One of them which must be considered by the Dialog Manager (DM) is the error propagation from ASR and NLU modules. In

<sup>1</sup>Through this paper, we will use the term *natural language* to include both *spoken language* and *written language*

general, errors in spoken dialog systems are prevalent due to errors in speech recognition or language understanding. These errors can cause the dialog system to misunderstand a user and in turn lead to an inappropriate response. To avoid these errors, a basic solution is to improve the accuracy and robustness of the recognition and understanding processes. However, it has been impossible to develop perfect ASR and NLU modules because of noisy environments and unexpected input. Therefore, the development of robust dialog management has also been one of the most important goals in research on practical spoken dialog systems.

In the dialog manager, a popular method to deal with these errors is to adopt dialog mechanisms for detecting and repairing potential errors at the conversational level (McTear et al., 2005; Torres et al., 2005; Lee et al., 2007). In human-computer communication, the goal of error recovery strategy is to maximize the user's satisfaction of using the system by guiding for the repair of the wrong information by human-computer interaction. On the other hand, there are different approaches to improve the robustness of dialog management using n-best hypotheses. Rather than Markov Decision Processes (MDPs), partially observable MDPs (POMDPs) potentially provide a much more powerful framework for robust dialog modeling since they consider n-best hypotheses to estimate the distribution of the belief state (Williams and Young, 2007).

In recent, we proposed another data-driven approach for the dialog modeling called Example-based Dialog Modeling (EBDM) (Lee et al., 2006a). However, difficulties occur when attempting to de-

ploy EBDM in practical spoken dialog systems in which ASR and NLU errors are frequent. Thus, this paper proposes a new method to improve the robustness of the EBDM framework using an agenda-based approach and n-best recognition hypotheses. We consider a domain-specific agenda to estimate the best dialog state and example because, in task-oriented systems, a current dialog state is highly correlated to the previous dialog state. We have also used the example-based error recovery approach to handle exceptional cases due to noisy input or unexpected focus shift.

This paper is organized as follows. Previous related work is described in Section 2, followed by the methodology and problems of the example-based dialog modeling in Section 3. An agenda-based approach for heuristics is presented in Section 4. Following that, we explain greedy selection with n-best hypotheses in Section 5. Section 6 describes the error recovery strategy to handle unexpected cases. Then, Section 7 provides the experimental results of a real user evaluation to verify our approach. Finally, we draw conclusions and make suggestions for future work in Section 8.

## 2 Related Work

In many spoken dialog systems that have been developed recently, various knowledge sources are used. One of the knowledge sources, which are usually application-dependent, is an agenda or task model. These are powerful representations for segmenting large tasks into more reasonable subtasks (Rich and Sidner, 1998; Bohus and Rudnicky, 2003; Young et al., 2007). These are manually designed for various purposes including dialog modeling, search space reduction, domain knowledge, and user simulation.

In Collagen (Rich and Sidner, 1998), a plan tree, which is an approximate representation of a partial SharedPlan, is composed of alternating act and plan recipe nodes for internal discourse state representation and discourse interpretation.

In addition, Bohus and Rudnicky (2003) have presented a RavenClaw dialog management which is an agenda-based architecture using hierarchical task decomposition and an expectation agenda. For modeling dialog, the domain-specific dialog control is represented in the *Dialog Task Specification* layer

using a tree of dialog agents, with each agent handling a certain subtask of the dialog task.

Recently, the problem of a large state space in POMDP framework has been solved by grouping states into partitions using user goal trees and ontology rules as heuristics (Young et al., 2007).

In this paper, we are interested in exploring algorithms that would integrate this knowledge source for users to achieve domain-specific goals. We used an agenda graph whose hierarchy reflects the natural order of dialog control. This graph is used to both keep track of the dialog state and to select the best example using multiple recognition hypotheses for augmenting previous EBDM framework.

## 3 Example-based Dialog Modeling

Our approach is implemented based on Example-Based Dialog Modeling (EBDM) which is one of generic dialog modelings. We begin with a brief overview of the EBDM framework in this section. EBDM was inspired by Example-Based Machine Translation (EBMT) (Nagao, 1984), a translation system in which the source sentence can be translated using similar example fragments within a large parallel corpus, without knowledge of the language's structure. The idea of EBMT can be extended to determine the next system actions by finding similar dialog examples within the dialog corpus. The system action can be predicted by finding semantically similar user utterances with the dialog state. The dialog state is defined as the set of relevant internal variables that affect the next system action. EBDM needs to automatically construct an example database from the dialog corpus. Dialog Example DataBase (DEDDB) is semantically indexed to generalize the data in which the indexing keys can be determined according to state variables chosen by a system designer for domain-specific applications (Figure 1). Each turn pair (user turn, system turn) in the dialog corpus is mapped to semantic instances in the DEDDB. The index constraints represent the state variables which are domain-independent attributes. To determine the next system action, there are three processes in the EBDM framework as follows:

- **Query Generation:** The dialog manager makes Structured Query Language (SQL)

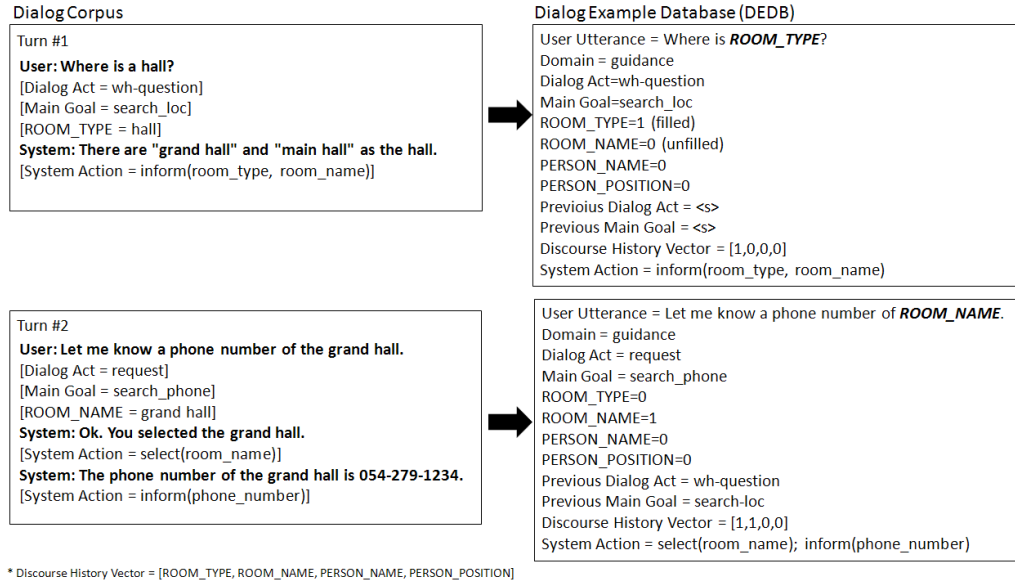


Figure 1: Indexing scheme for dialog example database on building guidance domain

statement using discourse history and NLU results.

- **Example Search:** The dialog manager searches for semantically similar dialog examples in the DEDB given the current dialog state. If no example is retrieved, some state variables can be ignored by relaxing particular variables according to the level of importance given the dialog's genre and domain.
- **Example Selection:** The dialog manager selects the best example to maximize the utterance similarity measure based on lexico-semantic similarity and discourse history similarity.

Figure 2 illustrates the overall strategy of EBDM framework for spoken dialog systems. The EBDM framework is a simple and powerful approach to rapidly develop natural language interfaces for multi-domain dialog processing (Lee et al., 2006b). However, in the context of spoken dialog system for domain-specific tasks, this framework must solve two problems: (1) Keeping track of the dialog state with a view to ensuring steady progress towards task completion, (2) Supporting n-best recognition hypotheses to improve the robustness of dialog manager. Consequently, we sought to solve these prob-

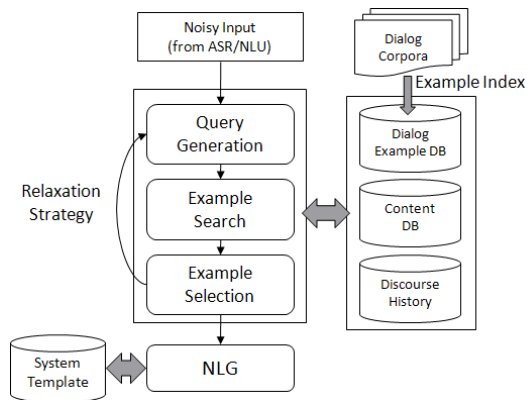


Figure 2: Strategy of the Example-Based Dialog Modeling (EBDM) framework.

lems by integrating the agenda graph as a heuristic which reflects the natural hierarchy and order of subtasks needed to complete the task.

## 4 Agenda Graph

In this paper, agenda graph  $G$  is simply a way of encoding the domain-specific dialog control to complete the task. An agenda is one of the subtask flows, which are possible paths from root node to terminal node.  $G$  is composed of nodes ( $v$ ) which correspond to possible intermediate steps in the process of completing the specified task, and edges ( $e$ ) which con-

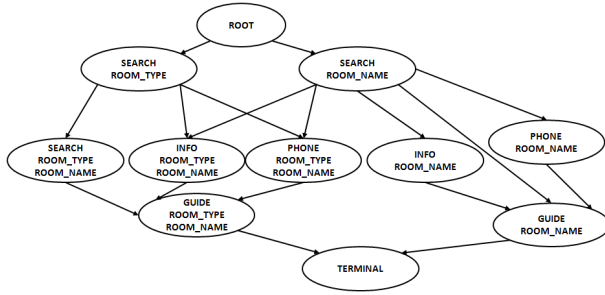


Figure 3: Example of an agenda graph for a building guidance.

nect nodes. In other words,  $v$  corresponds to user goal state to achieve domain-specific subtask in its expected agenda. Each node includes three different components: (1) A precondition that must be true before the subtask is executed; (2) A description of the node that includes its label and identifier; and (3) Links to nodes that will be executed at the subsequent turn. For every edge  $e_{ij} = (v_i, v_j)$ , we defined a transition probability based on prior knowledge of dialog flows. This probability can be assigned based on empirical analysis of human-computer conversations, assuming that the users behave in consistent, goal-directed ways. Alternatively, it can be assigned manually at the discretion of the system developer to control the dialog flow. This heuristic has advantages for practical spoken dialog system because a key condition for successful task-oriented dialog system is that the user and system know which task or subtask is currently being executed. To exemplify, Figure 3 illustrates part of the agenda graph for *PHOPE*, a building guidance robot using the spoken dialog system. In Figure 3,  $G$  is represented by a Directed Acyclic Graph (DAG), where each link in the graph reflects a transition between one user goal state and the next. The set of paths in  $G$  represent an agenda designed by the system developer. We adapted DAG representation because it is more intuitive and flexible than hierarchical tree representation. The syntax for graph representation in our system is described by an XML schema (Figure 4).

#### 4.1 Mapping Examples to Nodes

In the agenda graph  $G$ , each node  $v$  should hold relevant dialog examples corresponding to user goal states. Therefore, the dialog examples in DEDB are

```

<agenda_graph domain="guidance">
  <node node_id="2">
    <property>
      <label>Search Location with Room Type and Room Name</label>
    </property>
    <precondition>
      <main_goal>SEARCH_LOC</main_goal>
      <slot_status name="LOC_ROOM_NAME">1</slot_status>
      <slot_status name="LOC_ROOM_TYPE">1</slot_status>
      <slot_status name="LOC_ROOM_NUMBER">0</slot_status>
    </precondition>
    <next>
      <node_id prob="0.15">4</node_id>
      <node_id prob="0.25">5</node_id>
      <node_id prob="0.60">6</node_id>
    </next>
  </node>
</agenda_graph>
  
```

Figure 4: XML description for the agenda graph

mapped to a user goal state when a precondition of the node is true. Initially, the root node of the DAG is the starting state, where there is no dialog example. Then, the attributes of each dialog example are examined via the preconditions of each user goal node by breadth-first traversal. If the precondition is true, the node holds relevant that may appear in the user's goal state. The method of selecting the best of these examples will be described in 5.

#### 4.2 Discourse Interpretation

Inspired by Collagen (Rich and Sidner, 1998; Lesh et al., 2001), we investigated a discourse interpretation algorithm to consider how the current user's goal can contribute to the current agenda in a focus stack according to Lochbaum's discourse interpretation algorithm (Lochbaum, 1998). The focus stack takes into account the discourse structure by keeping track of discourse states. In our system, the focus stack is a set of user goal nodes which lead to completion of the subtask. The top on the focus stack is the previous node in this set. The focus stack is updated after every utterance. To interpret the type of the discourse state, this breaks down into five main cases of possible current node for an observed user's goal:

- *NEW\_TASK*: Starting a new task to complete a new agenda (Child of the root).
- *NEW\_SUB\_TASK*: Starting a new subtask to partially shift focus (A different child of the parent).

- *NEXT\_TASK*: Working on the next subtask contributing to current agenda (Its child node).
- *CURRENT\_TASK*: Repeating or modifying the observed goal on the current subtask (Current node).
- *PARENT\_TASK*: Modifying the observation on the previous subtask (Parent node).

Nodes in parentheses denote the topological position of the current node relative to the top node on the focus stack. If *NEXT\_TASK* is selected, the current node is pushed to the focus stack. *NEXT\_TASK* covers totally focused behavior, i.e., when there are no unexpected focus shifts. This occurs when the current user utterance is highly correlated to the previous system utterance. The remaining four cases cover various types of discourse state. For example, *NEW\_SUB\_TASK* involves starting a new subtask to partially shift focus, thereby popping the previous goal off the focus stack and pushing a new user goal for the new subtask. *NEW\_TASK*, which is placed on the node linked to root node, involves starting a new task to complete a new agenda. Therefore, a dialog is re-started and the current node is pushed onto the focus stack with the current user goal as its first element.

If none of the above cases holds, the discourse interpretation concludes that the current input should be rejected because we expect user utterances to be correlated to the previous turn in a task-oriented domain. Therefore, this interpretation does not contribute to the current agenda on the focus stack due to ASR and NLU errors that are due to noisy environments and unexpected input. These cases can be handled by using an error recovery strategy in Section 6.

Figure 5 shows some examples of pseudo-codes used in the discourse interpretation algorithm to select the best node among possible next nodes.  $S, H,$  and  $G$  denote the focus stack, hypothesis, and agenda graph, respectively. The **INTERPRET** algorithm is initially called to interpret the current discourse state. Furthermore, the essence of a discourse interpretation algorithm is to find candidate nodes of possible next subtask for an observed user goal, expressed in the definition of **GENERATE**. The **SELECT** algorithm selects the best node to maximize

<pre> INTERPRET(&lt;S,H&gt;,G) ≡ C ← GENERATE(&lt;S,H&gt;,G) if  C  = 1   then return discourse state in C else   c* ← SELECT(&lt;S,H&gt;,C)   return selected discourse state in c* </pre>	<pre> GENERATE(&lt;S,H&gt;,G) ≡ return a union set of : i) NEW_TASK(&lt;S,H&gt;,G) ii) NEW_SUB_TASK(&lt;S,H&gt;,G) iii) NEXT_TASK (&lt;S,H&gt;,G) iv) CURRENT_TASK(&lt;S,H&gt;,G) v) PARENT_TASK(&lt;S,H&gt;,G) </pre>
<pre> NEXT_TASK(&lt;S,H&gt;,G) ≡ C ← ∅ E ← retrieved examples of H foreach e ∈ E   foreach c<sub>s</sub> ∈ {children of top(S) G}     if e is an example of c<sub>s</sub>       then C ← C ∪ {c<sub>s</sub>} return C </pre>	<pre> SELECT(&lt;S,H&gt;,C) ≡ c* = argmax<sub>c</sub> ωS<sub>H</sub>(H)+(1-ω) S<sub>D</sub>(c ∈ C S) return c* </pre>

Figure 5: Pseudo-codes for the discourse interpretation algorithm

the score function based on current input and discourse structure given the focus stack. The details of how the score of candidate nodes are calculated are explained in Section 5.

## 5 Greedy Selection with n-best Hypotheses

Many speech recognizers can generate a list of plausible hypotheses (n-best list) but output only the most probable one. Examination of the n-best list reveals that the best hypothesis, the one with the lowest word error rate, is not always in top-1 position but sometimes in the lower rank of the n-best list. Therefore, we need to select the hypothesis that maximizes the scoring function among a set of n-best hypotheses of each utterance. The role of agenda graph is for a heuristic to score the discourse state to successfully complete the task given the focus stack.

The current system depends on a greedy policy which is based on immediate transitions rather than full transitions from the initial state. The greedy selection with n-best hypotheses is implemented as follows. Firstly, every hypothesis  $h_i$  is scanned and all possible nodes are generated using the discourse interpretation. Secondly, the multi-level score functions are computed for each candidate node  $c_i$  given a hypothesis  $h_i$ . Using the greedy algorithm, the node with the highest score is selected as the user goal state. Finally, the system actions are predicted by the dialog example to maximize the example score in the best node.

The generation of candidate nodes is based on multiple hypotheses from the previous EBDM

framework. This previous EBDM framework chose a dialog example to maximize the utterance similarity measure. However, our system generates a set of multiple dialog examples with each utterance similarity over a threshold given a specific hypothesis. Then, the candidate nodes are generated by matching to each dialog example bound to the node. If the number of matching nodes is exactly one, that node is selected. Otherwise, the best node which would be pushed onto the focus stack must be selected using multi-level score functions.

### 5.1 Node Selection

The node selection is determined by calculating some score functions. We defined multi-level score functions that combine the scores of ASR, SLU, and DM modules, which range from 0.00 to 1.00. The best node is selected by greedy search with multiple hypotheses  $H$  and candidate nodes  $C$  as follows:

$$c^* = \arg \max_{h_i \in H, c_i \in C} \omega S_H(h_i) + (1 - \omega) S_D(c_i|S)$$

where  $H$  is a list of n-best hypotheses and  $C$  is a set of nodes to be generated by the discourse interpretation. For the node selection, we divided the score function into two functions  $S_H(h_i)$ , hypothesis score, and  $S_D(c_i|S)$ , discourse score, where  $c_i$  is the focus node to be generated by single hypothesis  $h_i$ .

We defined the hypothesis score at the utterance level as

$$S_H(h_i) = \alpha S_{rec}(h_i) + \beta S_{cont}(h_i)$$

where  $S_{rec}(h_i)$  denotes the recognition score which is a generalized confidence score over the confidence score of the top-rank hypothesis.  $S_{cont}(h_i)$  is the content score in the view of content management to access domain-specific contents. For example, in the building guidance domain, these contents would be a building knowledge database including room name, room number, and room type. The score is defined as:

$$S_{cont}(h_i) = \begin{cases} \frac{N(C_{h_i})}{N(C_{prev})} & \text{if } C_{h_i} \subseteq C_{prev} \\ \frac{N(C_{h_i})}{N(C_{total})} & \text{if } C_{h_i} \not\subseteq C_{prev} \end{cases}$$

where  $C_{prev}$  is a set of contents at the previous turn and  $C_{total}$  is a set of total contents in the content

database.  $C_{h_i}$  denotes a set of focused contents by hypothesis  $h_i$  at the current turn.  $N(C)$  represents the number of contents  $C$ . This score reflects the degree of content coherence because the number of contents of interest has been gradually reduced without any unexpected focus shift. In the hypothesis score,  $\alpha$  and  $\beta$  denote weights which depend on the accuracy of speech recognition and language understanding, respectively.

In addition to the hypothesis score, we defined the discourse score  $S_D$  at the discourse level to consider the discourse structure between the previous node and current node given the focus stack  $S$ . This score is the degree to which candidate node  $c_i$  is in focus with respect to the previous user goal and system utterance. In the agenda graph  $G$ , each transition has its own probability as prior knowledge. Therefore, when  $c_i$  is *NEXT\_TASK*, the discourse score is computed as

$$S_D(c_i|S) = P(c_i|c = top(S))$$

where  $P(c_i|c = top(S))$  is a transition probability from the top node  $c$  on the focus stack  $S$  to the candidate node  $c_i$ . However, there is a problem for cases other than *NEXT\_TASK* because the graph has no backward probability. To solve this problem, we assume that the transition probability may be lower than that of the *NEXT\_TASK* case because a user utterance is likely to be influenced by the previous turn. Actually, when using the task-oriented dialog system, typical users stay focused most of the time during imperfect communication (Lesh et al., 2001). To assign the backward transition probability, we obtain the minimum transition probability  $P_{min}(S)$  among from the top node on the focus stack  $S$  to its children. Then, the discourse score  $S_D$  can be formalized when the candidate node  $c_i$  does not correspond to *NEXT\_TASK* as follows:

$$S_D(c_i|S) = \max\{P_{min}(S) - \lambda Dist(c_i, c), 0\}$$

where  $\lambda$  is a penalty of distance between candidate node and previous node,  $Dist(c_i, c)$ , according to type of candidate node such as *NEW\_TASK* and *NEW\_SUB\_TASK*. The simplest case is to uniformly assign  $\lambda$  to a specific value.

To select the best node using the node score, we use  $\omega$  ( $0 \leq \omega \leq 1$ ) as an interpolation weight

between the hypothesis score  $S_h$  and the discourse score  $S_D$ . This weight is empirically assigned according to the characteristics of the dialog genre and task. For example,  $\omega$  can set lower to manage the transactional dialog in which the user utterance is highly correlated to the previous system utterance, i.e., a travel reservation task, because this task usually has preference orders to fill slots.

## 5.2 Example Selection

After selecting the best node, we use the example score to select the best dialog example mapped into this node.

$$e^* = \arg \max_{e_j \in E(c^*)} \omega S_{utter}(h^*, e_j) + (1 - \omega) S_{sem}(h^*, e_j)$$

where  $h^*$  is the best hypothesis to maximize the node score and  $e_j$  is a dialog example in the best node  $c^*$ .  $S_{utter}(h, e_j)$  denotes the value of the utterance similarity of the user's utterances between the hypothesis  $h$  and dialog example  $e_j$  in the best node  $c^*$  (Lee et al., 2006a).

To augment the utterance similarity used in the EBDM framework, we also defined the semantic score for example selection,  $S_{sem}(h, e_j)$ :

$$S_{sem}(h, e_j) = \frac{\# \text{ of matching index keys}}{\# \text{ of total index keys}}$$

The semantic score is the ratio of matching index keys to the number of total index keys between hypothesis  $h$  and example record  $e_j$ . This score reflects that a dialog example is semantically closer to the current utterance if the example is selected with more index keys. After processing of the node and example selection, the best example is used to predict the system actions. Therefore, the dialog manager can predict the next actions with the agenda graph and n-best recognition hypotheses.

## 6 Error Recovery Strategy

As noted in Section 4.2, the discourse interpretation sometimes fails to generate candidate nodes. In addition, the dialog manager should confirm the current information when the score falls below some threshold. For these cases, we adapt an example-based error recovery strategy (Lee et al., 2007). In this approach, the system detects that something is

wrong in the user's utterance and takes immediate steps to address the problem using some help messages such as *UtterHelp*, *InfoHelp*, and *UsageHelp* in the example-based error recovery strategies. We also added a new help message, *AgendaHelp*, that uses the agenda graph and the label of each node to tell the user which subtask to perform next such as "SYSTEM: Next, you can do the subtask 1)Search Location with Room Name or 2)Search Location with Room Type".

## 7 Experiment & Result

First we developed the spoken dialog system for *PHOPE* in which an intelligent robot can provide information about buildings (i.e., room number, room location, room name, room type) and people (i.e., name, phone number, e-mail address, cellular phone number). If the user selects a specific room to visit, then the robot takes the user to the desired room. For this system, ten people used the WOZ method to collect a dialog corpus of about 500 utterances from 100 dialogs which were based on a set of pre-defined 10 subjects relating to domain-specific tasks. Then, we designed an agenda graph and integrated it into the EBDM framework.

In an attempt to quantify the impact of our approach, five Korean users participated in a preliminary evaluation. We provided them with pre-defined scenarios and asked them to collect test data from 50 dialogs, including about 150 utterances. After processing each dialog, the participants completed a questionnaire to assess their satisfaction with aspects of the performance evaluation. The speech recognition hypotheses are obtained by using the Hidden Markov model Toolkit (HTK) speech recognizer adapted to our application domain in which the word error rate (WER) is 21.03%. The results of the *Task Completion Rate* (TCR) are shown in Table 1. We explored the effects of our agenda-based approach with n-best hypotheses compared to the previous EBDM framework which has no agenda graph and supports only 1-best hypothesis.

Note that using 10-best hypotheses and the agenda graph increases the TCR from 84.0% to 90.0%, that is, 45 out of 50 dialogs were completed successfully. The average number of turns ( $\#AvgTurn$ ) to completion was also shorter, which



shows 4.35 turns per a dialog using the agenda graph and 10-best hypotheses. From these results, we conclude that the use of the n-best hypotheses with the agenda graph is helpful to improve the robustness of the EBDM framework against noisy inputs.

System	#AvgTurn	TCR (%)
1-best(-AG)	4.65	84.0
10-best(+AG)	4.35	90.0

Table 1: Task completion rate according to using the AG (Agenda Graph) and n-best hypotheses for n=1 and n=10.

## 8 Conclusion & Discussion

This paper has proposed a new agenda-based approach with n-best recognition hypotheses to improve the robustness of the Example-based Dialog Modeling (EBDM) framework. The agenda graph can be thought of as a hidden cost of applying our methodology. However, an explicit agenda is necessary to successfully achieve the purpose of using spoken dialog system. Our preliminary results indicate this fact that the use of agenda graph as heuristics can increase the TCR. In addition, our approach is robust to recognition errors because it maintains multiple hypotheses for each user utterance.

There are several possible subjects for further research on our approach. First, the optimal interpolation weights should be determined. This task will require larger dialog corpora by using user simulation. Second, the cost of designing the agenda graph should be reduced. We have focused on developing a system to construct this graph semi-automatically by applying dialog state clustering and utterance clustering to achieve hierarchical clustering of dialog examples. Finally, future work will include expanding our system to other applications, such as navigation systems for automobiles.

## Acknowledgement

This work was supported by grant No. RTI04-02-06 from the Regional Technology Innovation Program and by the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy (MOICE) of Korea.

## References

- Bohus, B. and Rudnicky A. 2003. RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. *Proceedings of the European Conference on Speech, Communication and Technology*, 597–600.
- Grosz, B.J. and Kraus, S. 1996. Collaborative Plans for Complex Group Action. *Artificial Intelligence*, 86(2):269–357.
- Lee, C., Jung, S., Eun, J., Jeong, M., and Lee, G.G. 2006. A Situation-based Dialogue Management using Dialogue Examples. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 69–72.
- Lee, C., Jung, S., Jeong, M., and Lee, G.G. 2006. Chat and Goal-oriented Dialog Together: A Unified Example-based Architecture for Multi-domain Dialog Management. *Proceedings of the IEEE Spoken Language Technology Workshop*, 194–197.
- Lee, C., Jung, S., and Lee, G.G. 2007. Example-based Error Recovery Strategy For Spoken Dialog System. *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, 538–543.
- Lesh, N., Rich, C., and Sidner, C. 2001. Collaborating with focused and unfocused users under imperfect communication. *Proceedings of the International Conference on User Modeling*, 63–74.
- Lochbaum, K.E. 1998. A Collaborative Planning Model of Intentional Structure. *Computational Linguistics*, 24(4):525–572.
- McTear, M., O’Neil, I., Hanna, P., and Liu, X. 2005. Handling errors and determining confirmation strategies-An object-based approach. *Speech Communication*, 45(3):249–269.
- Nagao, M. 1984. A Frame Work of a Mechanical Translation between Japanese and English by Analogy Principle. *Proceedings of the international NATO symposium on artificial and human intelligence*, 173–180.
- Rich, C. and Sidner, C.. 1998. Collagen: A Collaboration Agent for Software Interface Agents. *Journal of User Modeling and User-Adapted Interaction*, 8(3):315–350.
- Torres, F., Hurtado, L.F., Garcia, F., Sanchis, E., and Segarra, E. 2005. Error Handling in a Stochastic Dialog System through Confidence Measure. *Speech Communication*, 45(3):211–229.
- Williams, J.D. and Young, S. 2007. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech Language*, 21(2):393–422.
- Young, S., Schatzmann, J., Weilhammer, K., and Ye, H.. 2007. The Hidden Information State Approach to Dialog Management. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 149–152.

# Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz data: Bootstrapping and Evaluation

**Verena Rieser**

School of Informatics  
University of Edinburgh  
Edinburgh, EH8 9LW, GB  
vrieser@inf.ed.ac.uk

**Oliver Lemon**

School of Informatics  
University of Edinburgh  
Edinburgh, EH8 9LW, GB  
olemon@inf.ed.ac.uk

## Abstract

We address two problems in the field of automatic optimization of dialogue strategies: learning effective dialogue strategies when no initial data or system exists, and evaluating the result with real users. We use Reinforcement Learning (RL) to learn multimodal dialogue strategies by interaction with a simulated environment which is “bootstrapped” from small amounts of Wizard-of-Oz (WOZ) data. This use of WOZ data allows development of optimal strategies for domains where no working prototype is available. We compare the RL-based strategy against a supervised strategy which mimics the wizards’ policies. This comparison allows us to measure relative improvement over the training data. Our results show that RL significantly outperforms Supervised Learning when interacting in simulation as well as for interactions with real users. The RL-based policy gains on average 50-times more reward when tested in simulation, and almost 18-times more reward when interacting with real users. Users also subjectively rate the RL-based policy on average 10% higher.

## 1 Introduction

Designing a spoken dialogue system is a time-consuming and challenging task. A developer may spend a lot of time and effort anticipating the potential needs of a specific application environment and then deciding on the most appropriate system action (e.g. confirm, present items, . . .). One of the key advantages of statistical optimisation methods, such as Reinforcement Learning (RL), for dialogue

strategy design is that the problem can be formulated as a principled mathematical model which can be automatically trained on real data (Lemon and Pietquin, 2007; Frampton and Lemon, to appear). In cases where a system is designed from scratch, however, there is often no suitable in-domain data. Collecting dialogue data without a working prototype is problematic, leaving the developer with a classic chicken-and-egg problem.

We propose to learn dialogue strategies by simulation-based RL (Sutton and Barto, 1998), where the simulated environment is learned from small amounts of Wizard-of-Oz (WOZ) data. Using WOZ data rather than data from real Human-Computer Interaction (HCI) allows us to learn optimal strategies for domains where no working dialogue system already exists. To date, automatic strategy learning has been applied to dialogue systems which have already been deployed using hand-crafted strategies. In such work, strategy learning was performed based on already present extensive online operation experience, e.g. (Singh et al., 2002; Henderson et al., 2005). In contrast to this preceding work, our approach enables strategy learning in domains where no prior system is available. Optimised learned strategies are then available from the first moment of online-operation, and tedious hand-crafting of dialogue strategies is omitted. This independence from large amounts of in-domain dialogue data allows researchers to apply RL to new application areas beyond the scope of existing dialogue systems. We call this method ‘bootstrapping’.

In a WOZ experiment, a hidden human operator, the so called “wizard”, simulates (partly or com-

pletely) the behaviour of the application, while subjects are left in the belief that they are interacting with a real system (Fraser and Gilbert, 1991). That is, WOZ experiments only *simulate* HCI. We therefore need to show that a strategy bootstrapped from WOZ data indeed transfers to real HCI. Furthermore, we also need to introduce methods to learn useful user simulations (for training RL) from such limited data.

The use of WOZ data has earlier been proposed in the context of RL. (Williams and Young, 2004) utilise WOZ data to discover the state and action space for MDP design. (Prommer et al., 2006) use WOZ data to build a simulated user and noise model for simulation-based RL. While both studies show promising first results, their simulated environment still contains many hand-crafted aspects, which makes it hard to evaluate whether the success of the learned strategy indeed originates from the WOZ data. (Schatzmann et al., 2007) propose to ‘bootstrap’ with a simulated user which is entirely hand-crafted. In the following we propose an entirely data-driven approach, where all components of the simulated learning environment are learned from WOZ data. We also show that the resulting policy performs well for real users.

## 2 Wizard-of-Oz data collection

Our domains of interest are information-seeking dialogues, for example a multimodal in-car interface to a large database of music (MP3) files. The corpus we use for learning was collected in a multimodal study of German task-oriented dialogues for an in-car music player application by (Rieser et al., 2005). This study provides insights into natural methods of information presentation as performed by human wizards. 6 people played the role of an intelligent interface (the “wizards”). The wizards were able to speak freely and display search results on the screen by clicking on pre-computed templates. Wizards’ outputs were not restricted, in order to explore the different ways they intuitively chose to present search results. Wizard’s utterances were immediately transcribed and played back to the user with Text-To-Speech. 21 subjects (11 female, 10 male) were given a set of predefined tasks to perform, as well as a primary driving task, using a driving simu-

lator. The users were able to speak, as well as make selections on the screen. We also introduced artificial noise in the setup, in order to closer resemble the conditions of real HCI. Please see (Rieser et al., 2005) for further detail.

The corpus gathered with this setup comprises 21 sessions and over 1600 turns. Example 1 shows a typical multimodal presentation sub-dialogue from the corpus (translated from German). Note that the wizard displays quite a long list of possible candidates on an (average sized) computer screen, while the user is driving. This example illustrates that even for humans it is difficult to find an “optimal” solution to the problem we are trying to solve.

- (1) **User:** Please search for music by Madonna .  
**Wizard:** I found seventeen hundred and eleven items. The items are displayed on the screen.  
*[displays list]*  
**User:** Please select ‘Secret’.

For each session information was logged, e.g. the transcriptions of the spoken utterances, the wizard’s database query and the number of results, the screen option chosen by the wizard, and a rich set of contextual dialogue features was also annotated, see (Rieser et al., 2005).

Of the 793 wizard turns 22.3% were annotated as presentation strategies, resulting in 177 instances for learning, where the six wizards contributed about equal proportions.

Information about user preferences was obtained, using a questionnaire containing similar questions to the PARADISE study (Walker et al., 2000). In general, users report that they get distracted from driving if too much information is presented. On the other hand, users prefer shorter dialogues (most of the user ratings are negatively correlated with dialogue length). These results indicate that we need to find a strategy given the competing trade-offs between the number of results (large lists are difficult for users to process), the length of the dialogue (long dialogues are tiring, but collecting more information can result in more precise results), and the noise in the speech recognition environment (in high noise conditions accurate information is difficult to obtain). In the following we utilise the ratings from the user questionnaires to optimise a presentation strategy using simulation-based RL.

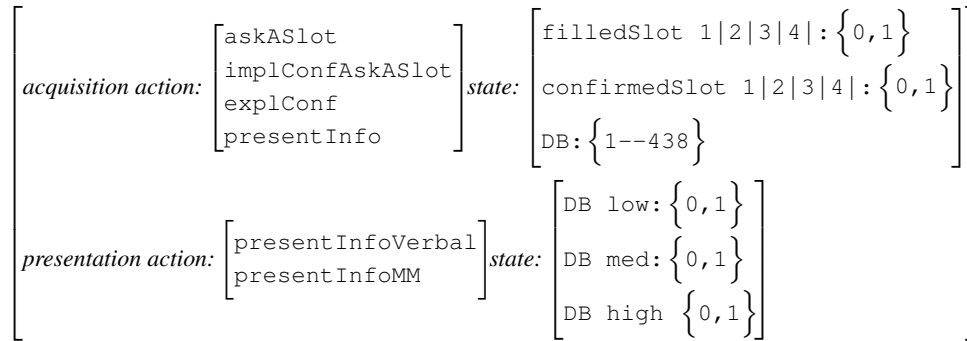


Figure 1: State-Action space for hierarchical Reinforcement Learning

### 3 Simulated Learning Environment

Simulation-based RL (also known as “model-free” RL) learns by interaction with a simulated environment. We obtain the simulated components from the WOZ corpus using data-driven methods. The employed database contains 438 items and is similar in retrieval ambiguity and structure to the one used in the WOZ experiment. The dialogue system used for learning comprises some obvious constraints reflecting the system logic (e.g. that only filled slots can be confirmed), implemented as Information State Update (ISU) rules. All other actions are left for optimisation.

#### 3.1 MDP and problem representation

The structure of an information seeking dialogue system consists of an information acquisition phase, and an information presentation phase. For information acquisition the task of the dialogue manager is to gather ‘enough’ search constraints from the user, and then, ‘at the right time’, to start the information presentation phase, where the presentation task is to present ‘the right amount’ of information in the right way— either on the screen or listing the items verbally. What ‘the right amount’ actually means depends on the application, the dialogue context, and the preferences of users. For optimising dialogue strategies information acquisition and presentation are two closely interrelated problems and need to be optimised simultaneously: *when* to present information depends on the available options for *how* to present them, and vice versa. We therefore formulate the problem as a Markov Decision Process (MDP), relating states to actions in a hierarchical manner (see Figure 1): 4 actions are available for

the information acquisition phase; once the action `presentInfo` is chosen, the information presentation phase is entered, where 2 different actions for output realisation are available. The state-space comprises 8 binary features representing the task for a 4 slot problem: `filledSlot` indicates whether a slot is filled, `confirmedSlot` indicates whether a slot is confirmed. We also add features that human wizards pay attention to, using the feature selection techniques of (Rieser and Lemon, 2006b). Our results indicate that wizards only pay attention to the number of retrieved items (DB). We therefore add the feature DB to the state space, which takes integer values between 1 and 438, resulting in  $2^8 \times 438 = 112,128$  distinct dialogue states. In total there are  $4^{112,128}$  theoretically possible policies for information acquisition.<sup>1</sup> For the presentation phase the DB feature is discretised, as we will further discuss in Section 3.6. For the information presentation phase there are  $2^{2^3} = 256$  theoretically possible policies.

#### 3.2 Supervised Baseline

We create a baseline by applying Supervised Learning (SL). This baseline mimics the average wizard behaviour and allows us to measure the relative improvements over the training data (cf. (Henderson et al., 2005)). For these experiments we use the WEKA toolkit (Witten and Frank, 2005). We learn with the decision tree J4.8 classifier, WEKA’s implementation of the C4.5 system (Quinlan, 1993), and rule induc-

<sup>1</sup>In practise, the policy space is smaller, as some of combinations are not possible, e.g. a slot cannot be confirmed before being filled. Furthermore, some incoherent action choices are excluded by the basic system logic.

	baseline	JRip	J48
timing	52.0( $\pm$ 2.2)	50.2( $\pm$ 9.7)	53.5( $\pm$ 11.7)
modality	51.0( $\pm$ 7.0)	93.5( $\pm$ 11.5)*	94.6( $\pm$ 10.0)*

Table 1: Predicted accuracy for presentation timing and modality (with standard deviation  $\pm$ ), \* denotes statistically significant improvement at  $p < .05$

tion JRIP, the WEKA implementation of RIPPER (Cohen, 1995). In particular, we learn models which predict the following wizard actions:

- Presentation timing: *when* the ‘average’ wizard starts the presentation phase
- Presentation modality: in *which modality* the list is presented.

As input features we use annotated dialogue context features, see (Rieser and Lemon, 2006b). Both models are trained using 10-fold cross validation. Table 1 presents the results for comparing the accuracy of the learned classifiers against the majority baseline. For presentation timing, none of the classifiers produces significantly improved results. Hence, we conclude that there is no distinctive pattern the wizards follow for *when* to present information. For strategy implementation we therefore use a frequency-based approach following the distribution in the WOZ data: in 0.48 of cases the baseline policy decides to present the retrieved items; for the rest of the time the system follows a hand-coded strategy. For learning presentation modality, both classifiers significantly outperform the baseline. The learned models can be rewritten as in Algorithm 1. Note that this rather simple algorithm is meant to represent the average strategy as present in the initial data (which then allows us to measure the relative improvements of the RL-based strategy).

---

**Algorithm 1** *SupervisedStrategy*

---

```

1: if  $DB \leq 3$  then
2:   return presentInfoVerbal
3: else
4:   return presentInfoMM
5: end if

```

---

### 3.3 Noise simulation

One of the fundamental characteristics of HCI is an error prone communication channel. Therefore, the simulation of channel noise is an important aspect of the learning environment. Previous work uses data-intensive simulations of ASR errors, e.g. (Pietquin and Dutoit, 2006). We use a simple model simulating the effects of non- and misunderstanding on the interaction, rather than the noise itself. This method is especially suited to learning from small data sets. From our data we estimate a 30% chance of user utterances to be misunderstood, and 4% to be complete non-understandings. We simulate the effects noise has on the user behaviour, as well as for the task accuracy. For the user side, the noise model defines the likelihood of the user accepting or rejecting the system’s hypothesis (for example when the system utters a confirmation), i.e. in 30% of the cases the user rejects, in 70% the user agrees. These probabilities are combined with the probabilities for user actions from the user simulation, as described in the next section. For non-understandings we have the user simulation generating Out-of-Vocabulary utterances with a chance of 4%. Furthermore, the noise model determines the likelihood of task accuracy as calculated in the reward function for learning. A filled slot which is not confirmed by the user has a 30% chance of having been mis-recognised.

### 3.4 User simulation

A user simulation is a predictive model of real user behaviour used for automatic dialogue strategy development and testing. For our domain, the user can either add information (`add`), repeat or paraphrase information which was already provided at an earlier stage (`repeat`), give a simple yes-no answer (`y/n`), or change to a different topic by providing a different slot value than the one asked for (`change`). These actions are annotated manually ( $\kappa = .7$ ). We build two different types of user simulations, one is used for strategy training, and one for testing. Both are simple bi-gram models which predict the next user action based on the previous system action ( $P(a_{user}|a_{system})$ ). We face the problem of learning such models when training data is sparse. For training, we therefore use a cluster-based user simulation method, see (Rieser

and Lemon, 2006a). For testing, we apply smoothing to the bi-gram model. The simulations are evaluated using the SUPER metric proposed earlier (Rieser and Lemon, 2006a), which measures variance and consistency of the simulated behaviour with respect to the observed behaviour in the original data set. This technique is used because for training we need more variance to facilitate the exploration of large state-action spaces, whereas for testing we need simulations which are more realistic. Both user simulations significantly outperform random and majority class baselines. See (Rieser, 2008) for further details.

### 3.5 Reward modelling

The reward function defines the goal of the overall dialogue. For example, if it is most important for the dialogue to be efficient, the reward penalises dialogue length, while rewarding task success. In most previous work the reward function is manually set, which makes it “the most hand-crafted aspect” of RL (Paek, 2006). In contrast, we learn the reward model from data, using a modified version of the PARADISE framework (Walker et al., 2000), following pioneering work by (Walker et al., 1998). In PARADISE multiple linear regression is used to build a predictive model of subjective user ratings (from questionnaires) from objective dialogue performance measures (such as dialogue length). We use PARADISE to predict Task Ease (a variable obtained by taking the average of two questions in the questionnaire)<sup>2</sup> from various input variables, via stepwise regression. The chosen model comprises dialogue length in turns, task completion (as manually annotated in the WOZ data), and the multimodal user score from the user questionnaire, as shown in Equation 2.

$$TaskEase = -20.2 * dialogueLength + 11.8 * taskCompletion + 8.7 * multimodalScore; \quad (2)$$

This equation is used to calculate the overall reward for the information acquisition phase. During learning, Task Completion is calculated online according to the noise model, penalising all slots which are filled but not confirmed.

<sup>2</sup>“The task was easy to solve.”, “I had *no* problems finding the information I wanted.”

For the information presentation phase, we compute a local reward. We relate the multimodal score (a variable obtained by taking the average of 4 questions)<sup>3</sup> to the number of items presented (DB) for each modality, using curve fitting. In contrast to linear regression, curve fitting does not assume a linear inductive bias, but it selects the most likely model (given the data points) by function interpolation. The resulting models are shown in Figure 3.5. The reward for multimodal presentation is a quadratic function that assigns a maximal score to a strategy displaying 14.8 items (curve inflection point). The reward for verbal presentation is a linear function assigning negative scores to all presented items  $\leq 4$ . The reward functions for information presentation intersect at no. items=3. A comprehensive evaluation of this reward function can be found in (Rieser and Lemon, 2008a).

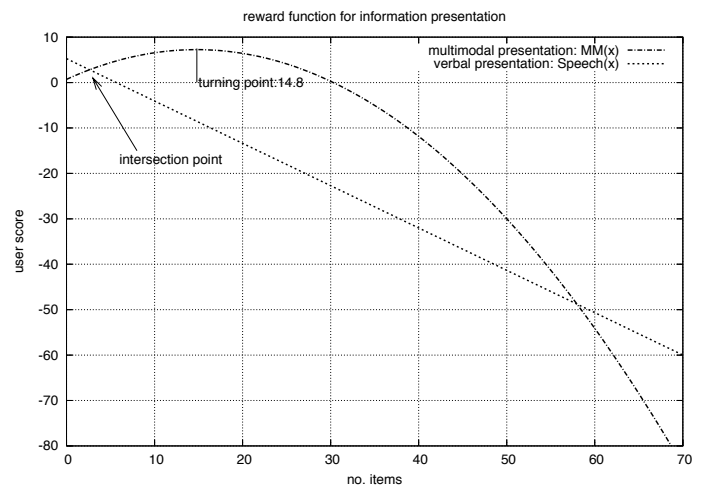


Figure 2: Evaluation functions relating number of items presented in different modalities to multimodal score

### 3.6 State space discretisation

We use linear function approximation in order to learn with large state-action spaces. Linear function approximation learns linear estimates for expected reward values of actions in states represented as feature vectors. This is inconsistent with the idea

<sup>3</sup>“I liked the combination of information being displayed on the screen and presented verbally.”, “Switching between modes did *not* distract me.”, “The displayed lists and tables contained on average the right amount of information.”, “The information presented verbally was easy to remember.”

of non-linear reward functions (as introduced in the previous section). We therefore quantise the state space for information presentation. We partition the database feature into 3 bins, taking the first intersection point between verbal and multimodal reward and the turning point of the multimodal function as discretisation boundaries. Previous work on learning with large databases commonly quantises the database feature in order to learn with large state spaces using manual heuristics, e.g. (Levin et al., 2000; Heeman, 2007). Our quantisation technique is more principled as it reflects user preferences for multi-modal output. Furthermore, in previous work database items were not only quantised in the state-space, but also in the reward function, resulting in a direct mapping between quantised retrieved items and discrete reward values, whereas our reward function still operates on the continuous values. In addition, the decision *when* to present a list (information acquisition phase) is still based on continuous DB values. In future work we plan to engineer new state features in order to learn with non-linear rewards while the state space is still continuous. A continuous representation of the state space allows learning of more fine-grained local trade-offs between the parameters, as demonstrated by (Rieser and Lemon, 2008b).

### 3.7 Testing the Learned Policies in Simulation

We now train and test the multimodal presentation strategies by interacting with the simulated learning environment. For the following RL experiments we used the REALL-DUDE toolkit of (Lemon et al., 2006b). The SHARSHA algorithm is employed for training, which adds hierarchical structure to the well known SARSA algorithm (Shapiro and Langley, 2002). The policy is trained with the cluster-based user simulation over 180k system cycles, which results in about 20k simulated dialogues. In total, the learned strategy has 371 distinct state-action pairs (see (Rieser, 2008) for details).

We test the RL-based and supervised baseline policies by running 500 test dialogues with a smoothed user simulation (so that we are not training and testing on the same simulation). We then compare quantitative dialogue measures performing a paired t-test. In particular, we compare mean values of the final rewards, number of filled and con-

firmed slots, dialog length, and items presented multimodally (`MM items`) and items presented verbally (`verbal items`). RL performs significantly better ( $p < .001$ ) than the baseline strategy. The only non-significant difference is the number of items presented verbally, where both RL and SL strategy settled on a threshold of less than 4 items. The mean performance measures for simulation-based testing are shown in Table 2 and Figure 3.

The major strength of the learned policy is that it learns to keep the dialogues reasonably short (on average 5.9 system turns for RL versus 8.4 turns for SL) by presenting lists as soon as the number of retrieved items is within tolerance range for the respective modality (as reflected in the reward function). The SL strategy in contrast has not learned the right timing nor an upper bound for displaying items on the screen. The results show that simulation-based RL with an environment bootstrapped from WOZ data allows learning of robust strategies which significantly outperform the strategies contained in the initial data set.

One major advantage of RL is that it allows us to provide additional information about user preferences in the reward function, whereas SL simply mimics the data. In addition, RL is based on delayed rewards, i.e. the optimisation of a final goal. For dialogue systems we often have measures indicating how successful and/or satisfying the overall performance of a strategy was, but it is hard to tell how things should have been exactly done in a specific situation. This is what makes RL specifically attractive for dialogue strategy learning. In the next section we test the learned strategy with real users.

## 4 User Tests

### 4.1 Experimental design

For the user tests the RL policy is ported to a working ISU-based dialogue system via table look-up, which indicates the action with the highest expected reward for each state (cf. (Singh et al., 2002)). The supervised baseline is implemented using standard threshold-based update rules. The experimental conditions are similar to the WOZ study, i.e. we ask the users to solve similar tasks, and use similar questionnaires. Furthermore, we decided to use typed user input rather than ASR. The use of text input

Measure	SL baseline		RL Strategy	
	SIM	REAL	SIM	REAL
av. turns	8.42( $\pm$ 3.04)	5.86( $\pm$ 3.2)	5.9( $\pm$ 2.4)***	5.07( $\pm$ 2.9)***
av. speech items	1.04( $\pm$ .2)	1.29( $\pm$ .4)	1.1( $\pm$ .3)	1.2( $\pm$ .4)
av. MM items	61.37( $\pm$ 82.5)	52.2( $\pm$ 68.5)	11.2( $\pm$ 2.4)***	8.73( $\pm$ 4.4)***
av. reward	-1741.3( $\pm$ 566.2)	-628.2( $\pm$ 178.6)	44.06( $\pm$ 51.5)***	37.62( $\pm$ 60.7)***

Table 2: Comparison of results obtained in simulation (SIM) and with real users (REAL) for SL and RL-based strategies; \*\*\* denotes significant difference between SL and RL at  $p < .001$

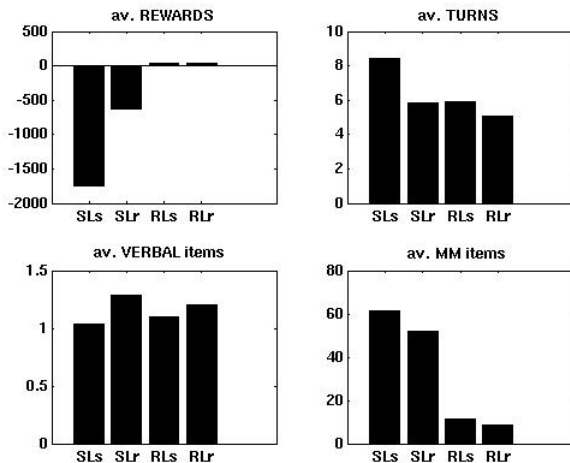


Figure 3: Graph comparison of objective measures: SLs = SL policy in simulation; SLr = SL policy with real users; RLs = RL policy in simulation; RLr = RL policy with real users.

allows us to target the experiments to the dialogue management decisions, and block ASR quality from interfering with the experimental results (Hajdinjak and Mihelic, 2006). 17 subjects (8 female, 9 male) are given a set of  $6 \times 2$  predefined tasks, which they solve by interaction with the RL-based and the SL-based system in controlled order. As a secondary task users are asked to count certain objects in a driving simulation. In total, 204 dialogues with 1,115 turns are gathered in this setup.

## 4.2 Results

In general, the users rate the RL-based significantly higher ( $p < .001$ ) than the SL-based policy. The results from a paired t-test on the user questionnaire

data show significantly improved Task Ease, better presentation timing, more agreeable verbal and multimodal presentation, and that more users would use the RL-based system in the future (Future Use). All the observed differences have a medium effects size ( $r \geq |.3|$ ).

We also observe that female participants clearly favour the RL-based strategy, whereas the ratings by male participants are more indifferent. Similar gender effects are also reported by other studies on multimodal output presentation, e.g. (Foster and Oberlander, 2006).

Furthermore, we compare objective dialogue performance measures. The dialogues of the RL strategy are significantly shorter ( $p < .005$ ), while fewer items are displayed ( $p < .001$ ), and the help function is used significantly less ( $p < .003$ ). The mean performance measures for testing with real users are shown in Table 2 and Figure 3. However, there is no significant difference for the performance of the secondary driving task.

## 5 Comparison of Results

We finally test whether the results obtained in simulation transfer to tests with real users, following (Lemon et al., 2006a). We evaluate the quality of the simulated learning environment by directly comparing the dialogue performance measures between simulated and real interaction. This comparison enables us to make claims regarding whether a policy which is ‘bootstrapped’ from WOZ data is transferable to real HCI. We first evaluate whether objective dialogue measures are transferable, using a paired t-test. For the RL policy there is no statistical difference in overall performance (reward), dialogue length (turns), and the number of presented items (verbal and multimodal items) between simulated



Measure	WOZ	SL	RL
av. Task Ease	.53±.14	.63±.26	.79±.21***
av. Future Use	.56±.16	.55±.21	.67±.20***

Table 3: Improved user ratings over the WOZ study where \*\*\* denotes  $p < .001$

and real interaction (see Table 2, Figure 3). This indicates that the learned strategy transfers well to real settings. For the SL policy the dialogue length for real users is significantly shorter than in simulation. From an error analysis we conclude that real users intelligently adapt to poor policies, e.g. by changing topic, whereas the simulated users do not react in this way.

Furthermore, we want to know whether the subjective user ratings for the RL strategy improved over the WOZ study. We therefore compare the user ratings from the WOZ questionnaire to the user ratings of the final user tests using a independent t-test and a Wilcoxon Signed Ranks Test. Users rate the RL-policy on average 10% higher. We are especially interested in the ratings for Task Ease (as this was the ultimate measure optimised with PARADISE) and Future Use, as we believe this measure to be an important indicator of acceptance of the technology. The results show that only the RL strategy leads to significantly improved user ratings (increasing average Task Ease by 49% and Future Use by 19%), whereas the ratings for the SL policy are not significantly better than those for the WOZ data, see Table 3.<sup>4</sup> This indicates that the observed difference is indeed due to the improved strategy (and not to other factors like the different user population or the embedded dialogue system).

## 6 Conclusion

We addressed two problems in the field of automatic optimization of dialogue strategies: learning effective dialogue strategies when no initial data or system exists, and evaluating the result with real users. We learned optimal strategies by interaction with a simulated environment which is bootstrapped from

<sup>4</sup>The ratings are normalised as some of the questions were on different scales.

a small amount of Wizard-of-Oz data, and we evaluated the result with real users. The use of WOZ data allows us to develop optimal strategies for domains where no working prototype is available. The developed simulations are entirely data driven and the reward function reflects real user preferences. We compare the Reinforcement Learning-based strategy against a supervised strategy which mimics the (human) wizards’ policies from the original data. This comparison allows us to measure relative improvement over the training data. Our results show that RL significantly outperforms SL in simulation as well as in interactions with real users. The RL-based policy gains on average 50-times more reward when tested in simulation, and almost 18-times more reward when interacting with real users. The human users also subjectively rate the RL-based policy on average 10% higher, and 49% higher for Task Ease. We also show that results obtained in simulation are comparable to results for real users. We conclude that a strategy trained from WOZ data via bootstrapping is transferable to real Human-Computer-Interaction.

In future work will apply similar techniques to statistical planning for Natural Language Generation in spoken dialogue (Lemon, 2008; Janarthanam and Lemon, 2008), (see the EC FP7 CLASSIC project: [www.classic-project.org](http://www.classic-project.org)).

## Acknowledgements

The research leading to these results has received funding from the European Community’s 7th Framework Programme (FP7/2007-2013) under grant agreement no. 216594 (CLASSIC project [www.classic-project.org](http://www.classic-project.org)), the EC FP6 project “TALK: Talk and Look, Tools for Ambient Linguistic Knowledge (IST 507802, [www.talk-project.org](http://www.talk-project.org)), from the EPSRC, project no. EP/E019501/1, and from the IRTG Saarland University.

## References

- W. W. Cohen. 1995. Fast effective rule induction. In *Proc. of the 12th ICML-95*.
- M. E. Foster and J. Oberlander. 2006. Data-driven generation of emphatic facial displays. In *Proc. of EACL*.
- M. Frampton and O. Lemon. (to appear). Recent research advances in Reinforcement Learning in Spoken Dialogue Systems. *Knowledge Engineering Review*.
- N. M. Fraser and G. N. Gilbert. 1991. Simulating speech systems. *Computer Speech and Language*, 5:81–99.
- M. Hajdinjak and F. Mihelic. 2006. The PARADISE evaluation framework: Issues and findings. *Computational Linguistics*, 32(2):263–272.
- P. Heeman. 2007. Combining reinforcement learning with information-state update rules. In *Proc. of NAACL*.
- J. Henderson, O. Lemon, and K. Georgila. 2005. Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data. In *Proc. of IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 68–75.
- S. Janarthanam and O. Lemon. 2008. User simulations for online adaptation and knowledge-alignment in Troubleshooting dialogue systems. In *Proc. of the 12th SEMDIAL Workshop (LONDial)*.
- O. Lemon and O. Pietquin. 2007. Machine learning for spoken dialogue systems. In *Proc. of Interspeech*.
- O. Lemon, K. Georgila, and J. Henderson. 2006a. Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation. In *Proc. of IEEE/ACL workshop on Spoken Language Technology (SLT)*.
- O. Lemon, X. Liu, D. Shapiro, and C. Tollander. 2006b. Hierarchical reinforcement learning of dialogue policies in a development environment for dialogue systems: REALL-DUDE. In *Proc. of the 10th SEMDIAL Workshop (BRANDial)*.
- O. Lemon. 2008. Adaptive Natural Language Generation in Dialogue using Reinforcement Learning. In *Proc. of the 12th SEMDIAL Workshop (LONDial)*.
- E. Levin, R. Pieraccini, and W. Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1).
- T. Paek. 2006. Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In *Proc. Dialog-on-Dialog Workshop, Interspeech*.
- O. Pietquin and T. Dutoit. 2006. A probabilistic framework for dialog simulation and optimal strategy learnin. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):589–599.
- T. Prommer, H. Holzapfel, and A. Waibel. 2006. Rapid simulation-driven reinforcement learning of multimodal dialog strategies in human-robot interaction. In *Proc. of Interspeech/ICSLP*.
- R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
- V. Rieser and O. Lemon. 2006a. Cluster-based user simulations for learning dialogue strategies. In *Proc. of Interspeech/ICSLP*.
- V. Rieser and O. Lemon. 2006b. Using machine learning to explore human multimodal clarification strategies. In *Proc. of ACL*.
- V. Rieser and O. Lemon. 2008a. Automatic learning and evaluation of user-centered objective functions for dialogue system optimisation. In *LREC*.
- V. Rieser and O. Lemon. 2008b. Does this list contain what you were searching for? Learning adaptive dialogue strategies for interactive question answering. *Journal of Natural Language Engineering (special issue on Interactive Question answering, to appear)*.
- V. Rieser, I. Kruijff-Korbayová, and O. Lemon. 2005. A corpus collection and annotation framework for learning multimodal clarification strategies. In *Proc. of the 6th SIGdial Workshop*.
- V. Rieser. 2008. *Bootstrapping Reinforcement Learning-based Dialogue Strategies from Wizard-of-Oz data (to appear)*. Ph.D. thesis, Saarland University.
- J. Schatzmann, B. Thomson, K. Weillhammer, H. Ye, and S. Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Proc. of HLT/NAACL*.
- D. Shapiro and P. Langley. 2002. Separating skills from preference: Using learning to program by reward. In *Proc. of the 19th ICML*.
- S. Singh, D. Litman, M. Kearns, and M. Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *JAIR*, 16.
- R. Sutton and A. Barto. 1998. *Reinforcement Learning*. MIT Press.
- M. Walker, J. Fromer, and S. Narayanan. 1998. Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *Proceedings of ACL/COLING*.
- M. Walker, C. Kamm, and D. Litman. 2000. Towards developing general models of usability with PARADISE. *Journal of Natural Language Engineering*, 6(3).
- J. Williams and S. Young. 2004. Using Wizard-of-Oz simulations to bootstrap reinforcement-learning-based dialog management systems. In *Proc. of the 4th SIGdial Workshop*.
- I. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques (2nd Edition)*. Morgan Kaufmann.

# Phrase Chunking using Entropy Guided Transformation Learning

Ruy L. Milidiú

Departamento de Informática  
PUC-Rio

Rio de Janeiro, Brazil  
milidiu@inf.puc-rio.br

Cícero Nogueira dos Santos

Departamento de Informática  
PUC-Rio

nogueira@inf.puc-rio.br

Julio C. Duarte

Centro Tecnológico do Exército  
Rio de Janeiro, Brazil

jduarte@ctex.eb.br

## Abstract

Entropy Guided Transformation Learning (ETL) is a new machine learning strategy that combines the advantages of decision trees (DT) and Transformation Based Learning (TBL). In this work, we apply the ETL framework to four phrase chunking tasks: Portuguese noun phrase chunking, English base noun phrase chunking, English text chunking and Hindi text chunking. In all four tasks, ETL shows better results than Decision Trees and also than TBL with hand-crafted templates. ETL provides a new training strategy that accelerates transformation learning. For the English text chunking task this corresponds to a factor of five speedup. For Portuguese noun phrase chunking, ETL shows the best reported results for the task. For the other three linguistic tasks, ETL shows state-of-the-art competitive results and maintains the advantages of using a rule based system.

## 1 Introduction

Phrase Chunking is a Natural Language Processing (NLP) task that consists in dividing a text into syntactically correlated parts of words. These phrases are non-overlapping, i.e., a word can only be a member of one chunk (Sang and Buchholz, 2000). It provides a key feature that helps on more elaborated NLP tasks such as parsing and information extraction.

Since the last decade, many high-performance chunking systems were proposed, such as, SVM-based (Kudo and Matsumoto, 2001; Wu et al.,

2006), Winnow (Zhang et al., 2002), voted-perceptrons (Carreras and Màrquez, 2003), Transformation-Based Learning (TBL) (Ramshaw and Marcus, 1999; Megyesi, 2002) and Hidden Markov Model (HMM) (Molina and Pla, 2002), Memory-based (Sang, 2002). State-of-the-art systems for English base noun phrase chunking and text chunking are based in statistical techniques (Kudo and Matsumoto, 2001; Wu et al., 2006; Zhang et al., 2002).

TBL is one of the most accurate rule-based techniques for phrase chunking tasks (Ramshaw and Marcus, 1999; Ngai and Florian, 2001; Megyesi, 2002). On the other hand, TBL rules must follow patterns, called templates, that are meant to capture the relevant feature combinations. The process of generating good templates is highly expensive. It strongly depends on the problem expert skills to build them. Even when a template set is available for a given task, it may not be effective when we change from a language to another (dos Santos and Oliveira, 2005).

In this work, we apply Entropy Guided Transformation Learning (ETL) for phrase chunking. ETL is a new machine learning strategy that combines the advantages of Decision Trees (DT) and TBL (dos Santos and Milidiú, 2007a). The ETL key idea is to use decision tree induction to obtain feature combinations (templates) and then use the TBL algorithm to generate transformation rules. ETL produces transformation rules that are more effective than decision trees and also eliminates the need of a problem domain expert to build TBL templates.

We evaluate the performance of ETL over four

phrase chunking tasks: (1) English Base Noun Phrase (NP) chunking; (2) Portuguese NP chunking; (3) English Text Chunking; and (4) Hindi Text Chunking. Base NP chunking consists in recognizing non-overlapping text segments that contain NPs. Text chunking consists in dividing a text into syntactically correlated parts of words. For these four tasks, ETL shows state-of-the-art competitive results and maintains the advantages of using a rule based system.

The remainder of the paper is organized as follows. In section 2, the ETL strategy is described. In section 3, the experimental design and the corresponding results are reported. Finally, in section 4, we present our concluding remarks.

## 2 Entropy Guided Transformation Learning

Entropy Guided Transformation Learning (ETL) is a new machine learning strategy that combines the advantages of Decision Trees (DT) and Transformation-Based Learning (TBL) (dos Santos and Milidiú, 2007a). The key idea of ETL is to use decision tree induction to obtain templates. Next, the TBL strategy is used to generate transformation rules. The proposed method is illustrated in the Fig. 1.

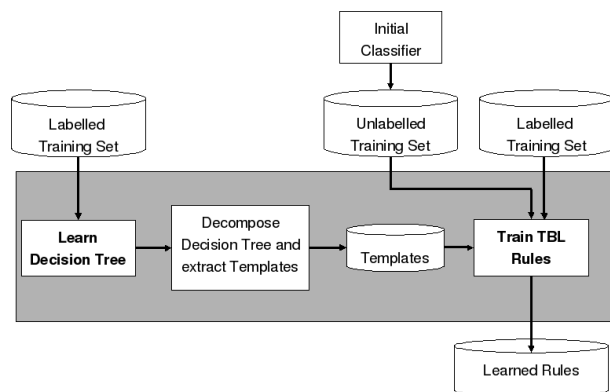


Figure 1: ETL - Entropy Guided Transformation Learning.

A combination of DT and TBL is presented in (Corston-Oliver and Gamon, 2003). The main difference between Corston-Oliver & Gamon work and the ETL strategy is that they extract candidate rules directly from the DT, and then use the TBL strategy

to select the appropriate rules. Another difference is that they use a binary DT, whereas ETL uses a DT that is not necessarily binary.

An evolutionary approach based on Genetic Algorithms (GA) to automatically generate TBL templates is presented in (Milidiú et al., 2007). Using a simple genetic coding, the generated template sets have efficacy near to the handcrafted templates for the tasks: English Base Noun Phrase Identification, Text Chunking and Portuguese Named Entities Recognition. The main drawback of this strategy is that the GA step is computationally expensive. If we need to consider a large context window or a large number of features, it can be infeasible.

The remainder of this section is organized as follows. In section 2.1, we describe the DT learning algorithm. In section 2.2, the TBL algorithm is depicted. In section 2.3, we depict the process of obtaining templates from a decision tree decomposition. Finally, in section 2.4, we present a *template evolution scheme* that speeds up the TBL step.

### 2.1 Decision Trees

Decision tree learning is one of the most widely used machine learning algorithms. It performs a partitioning of the training set using principles of Information Theory. The learning algorithm executes a general to specific search of a feature space. The most informative feature is added to a tree structure at each step of the search. Information Gain Ratio, which is based on the data Entropy, is normally used as the informativeness measure. The objective is to construct a tree, using a minimal set of features, that efficiently partitions the training set into classes of observations. After the tree is grown, a pruning step is carried out in order to avoid overfitting.

One of the most used algorithms for induction of a DT is the C4.5 (Quinlan, 1993). We use Quinlan's C4.5 system throughout this work.

### 2.2 Transformation-Based Learning

Transformation Based error-driven Learning (TBL) is a successful machine learning algorithm introduced by Eric Brill (Brill, 1995). It has since been used for several Natural Language Processing tasks, such as part-of-speech (POS) tagging (Brill, 1995), English text chunking (Ramshaw and Marcus, 1999; dos Santos and Milidiú, 2007b), spelling correc-

tion (Mangu and Brill, 1997), Portuguese appositive extraction (Freitas et al., 2006), Portuguese named entity extraction (Milidiú et al., 2006) and Portuguese noun-phrase chunking (dos Santos and Oliveira, 2005), achieving state-of-the-art performance in many of them.

TBL uses an error correcting strategy. Its main scheme is to generate an ordered list of rules that correct classification mistakes in the training set, which have been produced by an initial classifier.

The requirements of the algorithm are:

- two instances of the training set, one that has been correctly labeled, and another that remains unlabeled;
- an initial classifier, the *baseline system*, which classifies the unlabeled training set by trying to apply the correct class for each sample. In general, the baseline system is based on simple statistics of the labeled training set; and
- a set of rule templates, which are meant to capture the relevant feature combinations that would determine the sample's classification. Concrete rules are acquired by instantiation of this predefined set of rule templates.
- a threshold value, that is used as a stopping criteria for the algorithm and is needed to avoid overfitting to the training data.

The learning method is a mistake-driven greedy procedure that iteratively acquires a set of transformation rules. The TBL algorithm can be depicted as follows:

1. Starts applying the baseline system, in order to guess an initial classification for the unlabeled version of the training set;
2. Compares the resulting classification with the correct one and, whenever a classification error is found, all the rules that can correct it are generated by instantiating the templates. This template instantiation is done by capturing some contextual data of the sample being corrected. Usually, a new rule will correct some errors, but will also generate some other errors by changing correctly classified samples;

3. Computes the rules' scores (errors repaired - errors created). If there is not a rule with a score above an arbitrary threshold, the learning process is stopped;
4. Selects the best scoring rule, stores it in the set of learned rules and applies it to the training set;
5. Returns to step 2.

When classifying a new sample item, the resulting sequence of rules is applied according to its generation order.

### 2.3 DT Template Extraction

There are many ways to extract feature combinations from decision trees. In an path from the root to the leaves, more informative features appear first. Since we want to generate the most promising templates only, we just combine the more informative ones.

The process we use to extract templates from a DT includes a depth-first traversal of the DT. For each visited node, we create a new template that combines its parent node template with the feature used to split the data at that node. This is a very simple decomposition scheme. Nevertheless, it results into extremely effective templates. We also use pruned trees in all experiments shown in section 3.

Fig. 2 shows an excerpt of a DT generated for the English text chunking task<sup>1</sup>. Using the described method to extract templates from the DT shown in Fig. 2, we obtain the template set listed in the left side of Table 1. In order to generate more feature combinations, without largely increasing the number of templates, we extend the template set by including templates that do not have the root node feature. The extended template set for the DT shown in Fig. 2 is listed in the right side of the Table 1.

We have also tried some other strategies that extract a larger number of templates from a DT. However, the efficacy of the learned rules is quite similar to the one generated by the first method. This reinforces the conjecture that a DT generates informative feature combinations.

<sup>1</sup>CK[0] = Chunk tag of the current word (initial classifier result); CK[-1] = previous word Chunk tag; CK[1] = next word Chunk tag; POS[0] = current word POS tag; WRD[0] = current word.

Table 1: Text chunking DT Template set example

Template set	Extended template set	
CK[0]	CK[0]	
CK[0] CK[1]	CK[0] CK[1]	CK[1]
CK[0] CK[1] WRD[0]	CK[0] CK[1] WRD[0]	CK[1] WRD[0]
CK[0] CK[1] WRD[0] CK[-1]	CK[0] CK[1] WRD[0] CK[-1]	CK[1] WRD[0] CK[-1]
CK[0] CK[1] POS[0]	CK[0] CK[1] POS[0]	CK[1] POS[0]
CK[0] CK[-1]	CK[0] CK[-1]	CK[-1]

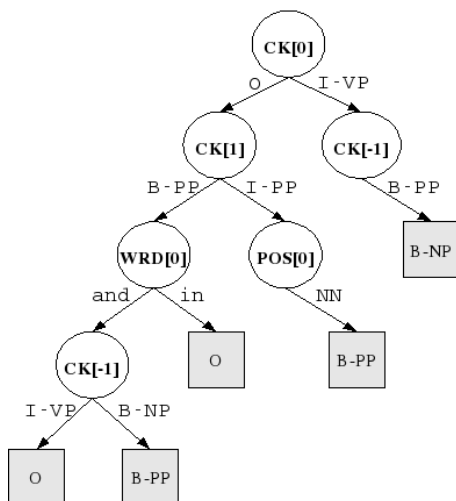


Figure 2: Text chunking decision tree excerpt.

## 2.4 Template Evolution Speedup

TBL training time is highly sensitive to the number and complexity of the applied templates. In (Curran and Wong, 2000), it is argued that we can better tune the *training time vs. templates complexity* trade-off by using an evolutionary template approach. The main idea is to apply only a small number of templates that evolve throughout the training. When training starts, templates are short, consisting of few feature combinations. As training proceeds, templates evolve to more complex ones that contain more feature combinations. In this way, only a few templates are considered at any point in time. Nevertheless, the descriptive power is not significantly reduced.

The template evolution approach can be easily implemented by using template sets extracted from a DT. We implement this idea by successively training TBL models. Each model uses only the templates

that contain feature combinations up to a given tree level. For instance, using the tree shown in Fig. 2, we have the following template sets for the three first training rounds<sup>2</sup>:

1. CK[0] CK[1];  
CK[0] CK[-1]
2. CK[0] CK[1] WRD[0];  
CK[0] CK[1] POS[0]
3. CK[0] CK[1] WRD[0] CK[-1]

Using the template evolution strategy, the training time is decreased by a factor of five for the English text chunking task. This is a remarkable reduction, since we use an implementation of the *fastTBL* algorithm (Ngai and Florian, 2001) that is already a very fast TBL version. The efficacy of the rules generated by the sequential training is quite similar to the one obtained by training with all the templates at the same time.

## 3 Experiments

This section presents the experimental setup and results of the application of ETL to four phrase chunking tasks. ETL results are compared with the results of DT and TBL using hand-crafted templates.

In the TBL step, for each one of the four chunking tasks, the initial classifier assigns to each word the chunk tag that was most frequently associated with the part-of-speech of that word in the training set.

The DT learning works as a feature selector and is not affected by irrelevant features. We have tried several context window sizes when training the classifiers. Some of the tested window sizes would be very hard to be explored by a domain expert using

<sup>2</sup>We ignore templates composed of only one feature test.

TBL alone. The corresponding huge number of possible templates would be very difficult to be managed by a template designer.

For the four tasks, the following experimental setup provided us our best results.

ETL in the ETL learning, we use the features *word*, *POS* and *chunk*. In order to overcome the sparsity problem, we only use the 200 most frequent words to induce the DT. In the DT learning, the chunk tag of the word is the one applied by the initial classifier. On the other hand, the chunk tag of neighbor words are the true ones. We report results for ETL trained with all the templates at the same time as well as using template evolution.

TBL the results for the TBL approach refers to TBL trained with the set of templates proposed in (Ramshaw and Marcus, 1999).

DT the best result for the DT classifier is shown. The features *word*, *POS* and *chunk* are used to generate the DT classifier. The chunk tag of a word and its neighbors are the ones guessed by the initial classifier. Using only the 100 most frequent words gives our best results.

In all experiments, the term  $WS=X$  subscript means that a window of size  $X$  was used for the given model. For instance,  $ETL_{WS=3}$  corresponds to ETL trained with window of size three, that is, the current token, the previous and the next one.

### 3.1 Portuguese noun phrase chunking

For this task, we use the SNR-CLIC corpus described in (Freitas et al., 2005). This corpus is tagged with both POS and NP tags. The NP tags are: I, for in NP; O, for out of NP; and B for the leftmost word of an NP beginning immediately after another NP. We divided the corpus into 3514-sentence (83346 tokens) training set and a 878-sentence (20798 tokens) test set.

In Table 2 we compare the results<sup>3</sup> of ETL with DT and TBL. We can see that ETL, even with a small window size, produces better results than DT and TBL. The  $F_{\beta=1}$  of the  $ETL_{WS=7}$  classifier is 1.8% higher than the one of TBL and 2.6% higher than the one of the DT classifier.

<sup>3</sup>#T = Number of templates.

Table 2: Portuguese noun phrase chunking.

	Acc. (%)	Prec. (%)	Rec. (%)	$F_{\beta=1}$ (%)	# T
BLS	96.57	62.69	74.45	68.06	–
$DT_{WS=13}$	97.35	83.96	87.27	85.58	–
TBL	97.45	85.48	87.32	86.39	100
$ETL_{WS=3}$	97.61	86.12	87.24	86.67	21
$ETL_{WS=5}$	97.68	86.85	87.49	87.17	35
$ETL_{WS=7}$	<b>97.82</b>	<b>88.15</b>	88.20	<b>88.18</b>	34
$ETL_{WS=9}$	<b>97.82</b>	88.02	<b>88.34</b>	<b>88.18</b>	40

Table 3 shows the results<sup>4</sup> of ETL using template evolution. As we can see, for the task of Portuguese noun phrase chunking, the template evolution strategy reduces the average training time in approximately 35%. On the other hand, there is a decrease of the classifier efficacy in some cases.

Table 3: Portuguese noun phrase chunking using ETL with template evolution.

	Acc. (%)	Prec. (%)	Rec. (%)	$F_{\beta=1}$ (%)	TTR (%)
$ETL_{WS=3}$	97.61	86.22	87.27	86.74	20.7
$ETL_{WS=5}$	97.56	86.39	87.10	86.74	38.2
$ETL_{WS=7}$	97.69	87.35	87.89	87.62	37.0
$ETL_{WS=9}$	<b>97.76</b>	<b>87.55</b>	<b>88.14</b>	<b>87.85</b>	41.9

In (dos Santos and Oliveira, 2005), a special set of six templates is shown. These templates are designed to reduce classification errors of preposition within the task of Portuguese noun phrase chunking. These templates use very specific domain knowledge and are difficult to DT and TBL to extract. Table 4 shows the results of an experiment where we include these six templates into the Ramshaw&Marcus template set and also into the template sets generated by ETL. Again, ETL produces better results than TBL.

Table 5 shows the results of using a committee composed by the three best ETL classifiers. The classification is done by selecting the most popular tag among all the three committee members. The achieved  $F_{\beta=1}$ , 89.14% is the best one ever reported for the SNR-CLIC corpus.

<sup>4</sup>TTR = Training time reduction.

Table 4: Portuguese noun phrase chunking using six additional hand-crafted templates.

	Acc. (%)	Prec. (%)	Rec. (%)	$F_{\beta=1}$ (%)	# T
BLS	96.57	62.69	74.45	68.06	–
TBL	97.60	86.79	88.12	87.45	106
ETL <sub>WS=3</sub>	97.73	86.95	88.40	87.67	27
ETL <sub>WS=5</sub>	97.87	88.35	89.02	88.68	41
ETL <sub>WS=7</sub>	97.91	88.12	<b>89.22</b>	88.67	40
ETL <sub>WS=9</sub>	<b>97.93</b>	<b>88.53</b>	89.11	<b>88.82</b>	46

Table 5: Committee with the classifiers ETL<sub>WS=5</sub>, ETL<sub>WS=7</sub> and ETL<sub>WS=9</sub>, shown in Table 4.

	Results (%)
Accuracy	97.97
Precision	88.62
Recall	89.67
$F_{\beta=1}$	89.14

### 3.2 English base noun phrase chunking

The data used in the base NP chunking experiments is the one by Ramshaw & Marcus (Ramshaw and Marcus, 1999). This corpus contains sections 15-18 and section 20 of the Penn Treebank, and is predivided into 8936-sentence (211727 tokens) training set and a 2012-sentence (47377 tokens) test. This corpus is tagged with both POS and chunk tags.

Table 6 compares the results of ETL with DT and TBL for the base NP chunking. We can see that ETL, even using a small window size, produces better results than DT and TBL. The  $F_{\beta=1}$  of the ETL<sub>WS=9</sub> classifier is 0.87% higher than the one of TBL and 2.31% higher than the one of the DT classifier.

Table 7 shows the results of ETL using template evolution. The template evolution strategy reduces the average training time in approximately 62%. Differently from the Portuguese NP chunking, we observe an increase of the classifier efficacy in almost all the cases.

Table 8 shows the results of using a committee composed by the eight ETL classifiers reported in this section. Table 8 also shows the results for a committee of SVM models presented in (Kudo and Matsumoto, 2001). SVM’s results are the state-of-

Table 6: Base NP chunking.

	Acc. (%)	Prec. (%)	Rec. (%)	$F_{\beta=1}$ (%)	# T
BLS	94.48	78.20	81.87	79.99	–
DT <sub>WS=11</sub>	97.03	89.92	91.16	90.53	–
TBL	97.42	91.68	92.26	91.97	100
ETL <sub>WS=3</sub>	97.54	91.93	92.78	92.35	68
ETL <sub>WS=5</sub>	97.55	92.43	92.77	92.60	85
ETL <sub>WS=7</sub>	97.52	92.49	92.70	92.59	106
ETL <sub>WS=9</sub>	<b>97.63</b>	<b>92.62</b>	<b>93.05</b>	<b>92.84</b>	122

Table 7: Base NP chunking using ETL with template evolution.

	Acc. (%)	Prec. (%)	Rec. (%)	$F_{\beta=1}$ (%)	TTR (%)
ETL <sub>WS=3</sub>	97.58	92.07	92.74	92.41	53.9
ETL <sub>WS=5</sub>	<b>97.63</b>	<b>92.66</b>	<b>93.16</b>	<b>92.91</b>	57.9
ETL <sub>WS=7</sub>	97.61	92.56	93.04	92.80	65.1
ETL <sub>WS=9</sub>	97.59	92.50	93.01	92.76	69.4

the-art for the Base NP chunking task. On the other hand, using a committee of ETL classifiers, we produce very competitive results and maintain the advantages of using a rule based system.

Table 8: Base NP chunking using a committee of eight ETL classifiers.

	Accuracy (%)	Precision (%)	Recall (%)	$F_{\beta=1}$ (%)
ETL	97.72	92.87	93.34	93.11
SVM	–	<b>94.15</b>	<b>94.29</b>	<b>94.22</b>

### 3.3 English text chunking

The data used in the English text chunking experiments is the CoNLL-2000 corpus, which is described in (Sang and Buchholz, 2000). It is composed by the same texts as the Ramshaw & Marcus (Ramshaw and Marcus, 1999) corpus.

Table 9 compares the results of ETL with DTs and TBL for English text chunking. ETL, even using a small window size, produces better results than DTs and TBL. The  $F_{\beta=1}$  of the ETL<sub>WS=3</sub> classifier is 0.28% higher than the one of TBL and 2.17% higher than the one of the DT classifier. It is an interesting linguistic finding that the use of a window of size 3



(the current token, the previous token and the next token) provides the current best results for this task.

Table 9: English text Chunking.

	Acc. (%)	Prec. (%)	Rec. (%)	$F_{\beta=1}$ (%)	# T
BLS	77.29	72.58	82.14	77.07	–
DT <sub>WS=9</sub>	94.29	89.55	91.00	90.27	–
TBL	95.12	92.05	92.28	92.16	100
ETL <sub>WS=3</sub>	<b>95.24</b>	<b>92.32</b>	<b>92.56</b>	<b>92.44</b>	105
ETL <sub>WS=5</sub>	95.12	92.19	92.27	92.23	167
ETL <sub>WS=7</sub>	95.13	92.24	92.32	92.28	183
ETL <sub>WS=9</sub>	95.07	92.10	92.27	92.19	205

Table 10 shows the results of ETL using template evolution. The template evolution strategy reduces the average training time by approximately 81%. On the other hand, there is a small decrease of the classifier efficacy in all cases.

Table 10: English text chunking using ETL with template evolution.

	Acc. (%)	Prec. (%)	Rec. (%)	$F_{\beta=1}$ (%)	TTR (%)
ETL <sub>WS=3</sub>	<b>95.21</b>	<b>92.14</b>	<b>92.53</b>	<b>92.34</b>	77.2
ETL <sub>WS=5</sub>	94.98	91.84	92.25	92.04	80.8
ETL <sub>WS=7</sub>	95.03	91.89	92.28	92.09	83.0
ETL <sub>WS=9</sub>	95.01	91.87	92.21	92.04	84.5

Table 11 shows the results of using a committee composed by the eight ETL classifiers reported in this section. Table 11 also shows the results for a SVM model presented in (Wu et al., 2006). SVM’s results are the state-of-the-art for the Text chunking task. On the other hand, using a committee of ETL classifiers, we produce very competitive results and maintain the advantages of using a rule based system.

Table 11: English text Chunking using a committee of eight ETL classifiers.

	Accuracy (%)	Precision (%)	Recall (%)	$F_{\beta=1}$ (%)
ETL	95.50	92.63	92.96	92.79
SVM	–	<b>94.12</b>	<b>94.13</b>	<b>94.12</b>

Table 12 shows the results, broken down by chunk

type, of using a committee composed by the eight ETL classifiers reported in this section.

Table 12: English text chunking results, broken down by chunk type, for the ETL committee.

	Precision (%)	Recall (%)	$F_{\beta=1}$ (%)
ADJP	75.59	72.83	74.19
ADVP	82.02	79.56	80.77
CONJP	35.71	55.56	43.48
INTJ	00.00	00.00	00.00
LST	00.00	00.00	00.00
NP	92.90	93.08	92.99
PP	96.53	97.63	97.08
PRT	66.93	80.19	72.96
SBAR	86.50	85.05	85.77
VP	92.84	93.58	93.21
Overall	92.63	92.96	92.79

### 3.4 Hindi text chunking

The data used in the Hindi text chunking experiments is the SPSAL-2007 corpus, which is described in (Bharati and Mannem, 2007). This corpus is pre-divided into a 20000-tokens training set, a 5000-tokens development set and a 5000-tokens test set. This corpus is tagged with both POS and chunk tags.

To fairly compare our approach with the ones presented in the SPSAL-2007, the POS tags of the test corpus were replaced by the ones predicted by an ETL-based Hindi POS Tagger. The description of our ETL pos tagger is beyond the scope of this work. Since the amount of training data is very small (20000 tokens), the accuracy of the ETL Hindi POS tagger is low, 77.50% for the test set.

The results are reported in terms of chunking accuracy, the same performance measure used in the SPSAL-2007. Table 13 compares the results of ETL with DT and TBL for Hindi text chunking. ETL produces better results than DT and achieves the same performance of TBL using 60% less templates. We believe that ETL performance is not as good as in the other tasks mainly because of the small amount of training data, which increases the sparsity problem.

We do not use template evolution for Hindi text

chunking. Since the training corpus is very small, the training time reduction is not significant.

Table 13: Hindi text Chunking.

	Accuracy (%)	# Templates
BLS	70.05	–
DT <sub>WS=5</sub>	78.20	–
TBL	<b>78.53</b>	100
ETL <sub>WS=5</sub>	<b>78.53</b>	30

Table 14 compares the results of ETL with the two best Hindi text chunkers at SPSAL-2007 (Bharati and Mannem, 2007). The first one is a combination of Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (PVS and Gali, 2007). The second is based in Maximum Entropy Models (MaxEnt) (Dandapat, 2007). ETL performs better than MaxEnt and worst than HMM+CRF. It is important to note that the accuracy of the POS tagger used by (PVS and Gali, 2007) (78.66%) is better than ours (77.50%). The POS tagging quality directly affects the chunking accuracy.

Table 14: Comparison with best systems of SPSAL-2007

	Accuracy (%)
HMM + CRF	<b>80.97</b>
ETL <sub>WS=5</sub>	78.53
MaxEnt	74.92

## 4 Conclusions

In this paper, we approach the phrase chunking task using Entropy Guided Transformation Learning (ETL). We carry out experiments with four phrase chunking tasks: Portuguese noun phrase chunking, English base noun phrase chunking, English text chunking and Hindi text chunking. In all four tasks, ETL shows better results than Decision Trees and also than TBL with hand-crafted templates. ETL provides a new training strategy that accelerates transformation learning. For the English text chunking task this corresponds to a factor of five speedup. For Portuguese noun phrase chunking, ETL shows the best reported results for the task. For the other

three linguistic tasks, ETL shows competitive results and maintains the advantages of using a rule based system.

## References

- Akshar Bharati and Prashanth R. Mannem. 2007. Introduction to shallow parsing contest on south asian languages. In *Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL)*, pages 1–8.
- Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Comput. Linguistics*, 21(4):543–565.
- Xavier Carreras and Lluís Màrquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of RANLP-2003*, Borovets, Bulgaria.
- Simon Corston-Oliver and Michael Gamon. 2003. Combining decision trees and transformation-based learning to correct transferred linguistic representations. In *Proceedings of the Ninth Machine Translation Summit*, pages 55–62, New Orleans, USA. Association for Machine Translation in the Americas.
- J. R. Curran and R. K. Wong. 2000. Formalisation of transformation-based learning. In *Proceedings of the Australian Computer Science Conference - ACSC*, pages 51–57, Canberra, Australia.
- Sandipan Dandapat. 2007. Part of speech tagging and chunking with maximum entropy model. In *Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL)*, pages 29–32.
- Cícero N. dos Santos and Ruy L. Milidiú. 2007a. Entropy guided transformation learning. Technical Report 29/07, Departamento de Informática, PUC-Rio.
- Cícero N. dos Santos and Ruy L. Milidiú. 2007b. Probabilistic classifications with tbl. In *Proceedings of Eighth International Conference on Intelligent Text Processing and Computational Linguistics – CICLing*, pages 196–207, Mexico City, Mexico, February.
- Cícero N. dos Santos and Claudia Oliveira. 2005. Constrained atomic term: Widening the reach of rule templates in transformation based learning. In *EPIA*, pages 622–633.
- M. C. Freitas, M. Garrao, C. Oliveira, C. N. dos Santos, and M. Silveira. 2005. A anotação de um corpus para o aprendizado supervisionado de um modelo de sn. In *Proceedings of the III TIL / XXV Congresso da SBC*, São Leopoldo - RS - Brasil.
- M. C. Freitas, J. C. Duarte, C. N. dos Santos, R. L. Milidiú, R. P. Renteria, and V. Quental. 2006. A machine learning approach to the identification of appos-

- itives. In *Proceedings of Ibero-American AI Conference*, Ribeirão Preto, Brazil, October.
- T. Kudo and Y. Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the NAACL-2001*.
- Lidia Mangu and Eric Brill. 1997. Automatic rule acquisition for spelling correction. In *Proceedings of The Fourteenth International Conference on Machine Learning, ICML 97*. Morgan Kaufmann.
- Beáta Megyesi. 2002. Shallow parsing with pos taggers and linguistic features. *Journal of Machine Learning Research*, 2:639–668.
- Ruy L. Milidiú, Julio C. Duarte, and Roberto Cavalcante. 2006. Machine learning algorithms for portuguese named entity recognition. In *Proceedings of Fourth Workshop in Information and Human Language Technology (TIL'06)*, Ribeirão Preto, Brazil.
- Ruy L. Milidiú, Julio C. Duarte, and Cícero N. dos Santos. 2007. Tbl template selection: An evolutionary approach. In *Proceedings of Conference of the Spanish Association for Artificial Intelligence - CAEPIA*, Salamanca, Spain.
- Antonio Molina and Ferran Pla. 2002. Shallow parsing using specialized hmms. *J. Mach. Learn. Res.*, 2:595–613.
- Grace Ngai and Radu Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of North American ACL*, pages 40–47, June.
- Avinesh PVS and Karthik Gali. 2007. Part-of-speech tagging and chunking using conditional random fields and transformation based learning. In *Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL)*, pages 21–24.
- J. Ross Quinlan. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Lance Ramshaw and Mitch Marcus. 1999. Text chunking using transformation-based learning. In S. Armstrong, K.W. Church, P. Isabelle, S. Manzi, E. Tzoukermann, and D. Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*. Kluwer.
- Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th CONLL*, pages 127–132, Morristown, NJ, USA. Association for Computational Linguistics.
- Erik F. Tjong Kim Sang. 2002. Memory-based shallow parsing. *J. Mach. Learn. Res.*, 2:559–594.
- Yu-Chieh Wu, Chia-Hui Chang, and Yue-Shi Lee. 2006. A general and multi-lingual phrase chunking model based on masking method. In *Proceedings of 7th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 144–155.
- Tong Zhang, Fred Damerau, and David Johnson. 2002. Text chunking based on a generalization of winnow. *J. Mach. Learn. Res.*, 2:615–637.

# Learning Bigrams from Unigrams

Xiaojin Zhu<sup>†</sup> and Andrew B. Goldberg<sup>†</sup> and Michael Rabbat<sup>‡</sup> and Robert Nowak<sup>§</sup>

<sup>†</sup>Department of Computer Sciences, University of Wisconsin-Madison

<sup>‡</sup>Department of Electrical and Computer Engineering, McGill University

<sup>§</sup>Department of Electrical and Computer Engineering, University of Wisconsin-Madison

{jerryzhu, goldberg}@cs.wisc.edu, michael.rabbat@mcgill.ca, nowak@ece.wisc.edu

## Abstract

Traditional wisdom holds that once documents are turned into bag-of-words (unigram count) vectors, word orders are completely lost. We introduce an approach that, perhaps surprisingly, is able to learn a bigram language model from a set of bag-of-words documents. At its heart, our approach is an EM algorithm that seeks a model which maximizes the regularized marginal likelihood of the bag-of-words documents. In experiments on seven corpora, we observed that our learned bigram language models: i) achieve better test set perplexity than unigram models trained on the same bag-of-words documents, and are not far behind “oracle bigram models” trained on the corresponding ordered documents; ii) assign higher probabilities to sensible bigram word pairs; iii) improve the accuracy of ordered-document recovery from a bag-of-words. Our approach opens the door to novel phenomena, for example, privacy leakage from index files.

## 1 Introduction

A bag-of-words (BOW) is a basic document representation in natural language processing. In this paper, we consider a BOW in its simplest form, i.e., a *unigram count vector* or word histogram over the vocabulary. When performing the counting, word order is ignored. For example, the phrases “really neat” and “neat really” contribute equally to a BOW. Obviously, once a set of documents is turned into a set of BOWs, the word order information within them is completely lost—or is it?

In this paper, we show that one can in fact partly recover the order information. Specifically, *given a set of documents in unigram-count BOW representation, one can recover a non-trivial bigram language model (LM)*<sup>1</sup>, which has part of the power of a bigram LM trained on ordered documents. At first glance this seems impossible: How can one learn bigram information from unigram counts? However, we will demonstrate that *multiple* BOW documents enable us to recover some higher-order information.

Our results have implications in a wide range of natural language problems, in particular document privacy. With the wide adoption of natural language applications like desktop search engines, software programs are increasingly indexing computer users’ personal files for fast processing. Most index files include some variant of the BOW. As we demonstrate in this paper, if a malicious party gains access to BOW index files, it can recover more than just unigram frequencies: (i) the malicious party can recover a higher-order LM; (ii) with the LM it may attempt to recover the original ordered document from a BOW by finding the most-likely word permutation<sup>2</sup>. Future research will quantify the extent to which such a privacy breach is possible in theory, and will find solutions to prevent it.

There is a vast literature on language modeling; see, e.g., (Rosenfeld, 2000; Chen and Goodman, 1999; Brants et al., 2007; Roark et al., 2007). How-

<sup>1</sup>A trivial bigram LM is a unigram LM which ignores history:  $P(v|u) = P(v)$ .

<sup>2</sup>It is possible to use a generic higher-order LM, e.g., a trigram LM trained on standard English corpora, for this purpose. However, incorporating a user-specific LM helps.

ever, to the best of our knowledge, none addresses this reverse direction of learning higher-order LMs from lower-order data. This work is inspired by recent advances in inferring network structure from co-occurrence data, for example, for computer networks and biological pathways (Rabbat et al., 2007).

## 2 Problem Formulation and Identifiability

We assume that a vocabulary of size  $W$  is given. For notational convenience, we include in the vocabulary a special “begin-of-document” symbol  $\langle d \rangle$  which appears only at the beginning of each document. The training corpus consists of a collection of  $n$  BOW documents  $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ . Each BOW  $\mathbf{x}_i$  is a vector  $(x_{i1}, \dots, x_{iW})$  where  $x_{iu}$  is the number of times word  $u$  occurs in document  $i$ . Our goal is to learn a bigram LM  $\theta$ , represented as a  $W \times W$  transition matrix with  $\theta_{uv} = P(v|u)$ , from the BOW corpus. Note  $P(v|\langle d \rangle)$  corresponds to the initial state probability for word  $v$ , and  $P(\langle d \rangle|u) = 0, \forall u$ .

It is worth noting that traditionally one needs *ordered documents* to learn a bigram LM. A natural question that arises in our problem is whether or not a bigram LM can be recovered from the BOW corpus with any guarantee. Let  $\mathcal{X}$  denote the space of all possible BOWs. As a toy example, consider  $W = 3$  with the vocabulary  $\{\langle d \rangle, A, B\}$ . Assuming all documents have equal length  $|\mathbf{x}| = 4$  (including  $\langle d \rangle$ ), then  $\mathcal{X} = \{(\langle d \rangle:1, A:3, B:0), (\langle d \rangle:1, A:2, B:1), (\langle d \rangle:1, A:1, B:2), (\langle d \rangle:1, A:0, B:3)\}$ . Our training BOW corpus, when sufficiently large, provides the marginal distribution  $\hat{p}(\mathbf{x})$  for  $\mathbf{x} \in \mathcal{X}$ . Can we recover a bigram LM from  $\hat{p}(\mathbf{x})$ ?

To answer this question, we first need to introduce a generative model for the BOWs. We assume that the BOW corpus is generated from a bigram LM  $\theta$  in two steps: (i) An *ordered* document is generated from the bigram LM  $\theta$ ; (ii) The document’s unigram counts are collected to produce the BOW  $\mathbf{x}$ . Therefore, the probability of a BOW  $\mathbf{x}$  being generated by  $\theta$  can be computed by marginalizing over *unique orderings*  $\mathbf{z}$  of  $\mathbf{x}$ :

$$P(\mathbf{x}|\theta) = \sum_{\mathbf{z} \in \sigma(\mathbf{x})} P(\mathbf{z}|\theta) = \sum_{\mathbf{z} \in \sigma(\mathbf{x})} \prod_{j=2}^{|\mathbf{x}|} \theta_{z_{j-1}, z_j},$$

where  $\sigma(\mathbf{x})$  is the set of unique orderings, and  $|\mathbf{x}|$  is the document length. For example, if  $\mathbf{x} = (\langle d \rangle:1,$

A:2, B:1) then  $\sigma(\mathbf{x}) = \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}$  with  $\mathbf{z}_1 = \langle d \rangle A A B$ ,  $\mathbf{z}_2 = \langle d \rangle A B A$ ,  $\mathbf{z}_3 = \langle d \rangle B A A$ . Bigram LM recovery then amounts to finding a  $\theta$  that satisfies the system of marginal-matching equations

$$P(\mathbf{x}|\theta) = \hat{p}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X}. \quad (1)$$

As a concrete example where *one can exactly recover a bigram LM from BOWs*, consider our toy example again. We know there are only three free variables in our  $3 \times 3$  bigram LM  $\theta$ :  $r = \theta_{\langle d \rangle A}$ ,  $p = \theta_{AA}$ ,  $q = \theta_{BB}$ , since the rest are determined by normalization. Suppose the documents are generated from a bigram LM with true parameters  $r = 0.25, p = 0.9, q = 0.5$ . If our BOW corpus is very large, we will observe that 20.25% of the BOWs are  $(\langle d \rangle:1, A:3, B:0)$ , 37.25% are  $(\langle d \rangle:1, A:2, B:1)$ , and 18.75% are  $(\langle d \rangle:1, A:0, B:3)$ . These numbers are computed using the definition of  $P(\mathbf{x}|\theta)$ . We solve the reverse problem of finding  $r, p, q$  from the system of equations (1), now explicitly written as

$$\begin{cases} rp^2 = 0.2025 \\ rp(1-p) + r(1-p)(1-q) \\ \quad + (1-r)(1-q)p = 0.3725 \\ (1-r)q^2 = 0.1875. \end{cases}$$

The above system has only one valid solution, which is the correct set of bigram LM parameters  $(r, p, q) = (0.25, 0.9, 0.5)$ .

However, if the true parameters were  $(r, p, q) = (0.1, 0.2, 0.3)$  with proportions of BOWs being 0.4%, 19.8%, 8.1%, respectively, it is easy to verify that the system would have multiple valid solutions:  $(0.1, 0.2, 0.3)$ ,  $(0.8819, 0.0673, 0.8283)$ , and  $(0.1180, 0.1841, 0.3030)$ . In general, if  $\hat{p}(\mathbf{x})$  is known from the training BOW corpus, when can we *guarantee* to uniquely recover the bigram LM  $\theta$ ? This is the question of *identifiability*, which means the transition matrix  $\theta$  satisfying (1) exists and is unique. Identifiability is related to finding unique solutions of a system of polynomial equations since (1) is such a system in the elements of  $\theta$ . The details are beyond the scope of this paper, but applying the technique in (Basu and Boston, 2000), it is possible to show that for  $W = 3$  (including  $\langle d \rangle$ ) we need longer documents ( $|\mathbf{x}| \geq 5$ ) to ensure identifiability. The identifiability of more general cases is still an open research question.

### 3 Bigram Recovery Algorithm

In practice, the documents are not truly generated from a bigram LM, and the BOW corpus may be small. We therefore seek a maximum likelihood estimate of  $\theta$  or a regularized version of it. Equivalently, we no longer require equality in (1), but instead find  $\theta$  that makes the distribution  $P(\mathbf{x}|\theta)$  as close to  $\hat{p}(\mathbf{x})$  as possible. We formalize this notion below.

#### 3.1 The Objective Function

Given a BOW corpus  $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , its normalized log likelihood under  $\theta$  is  $\ell(\theta) \equiv \frac{1}{C} \sum_{i=1}^n \log P(\mathbf{x}_i|\theta)$ , where  $C = \sum_{i=1}^n (|\mathbf{x}_i| - 1)$  is the corpus length excluding  $\langle d \rangle$ 's. The idea is to find  $\theta$  that maximizes  $\ell(\theta)$ . This also brings  $P(\mathbf{x}|\theta)$  closest to  $\hat{p}(\mathbf{x})$  in the KL-divergence sense. However, to prevent overfitting, we regularize the problem so that  $\theta$  prefers to be close to a ‘‘prior’’ bigram LM  $\phi$ . The prior  $\phi$  is also estimated from the BOW corpus, and is discussed in Section 3.4. We define the regularizer to be an asymmetric dissimilarity  $\mathcal{D}(\phi, \theta)$  between the prior  $\phi$  and the learned model  $\theta$ . The dissimilarity is 0 if  $\theta = \phi$ , and increases as they diverge. Specifically, the KL-divergence between two word distributions conditioned on the same history  $u$  is  $KL(\phi_u \parallel \theta_u) = \sum_{v=1}^W \phi_{uv} \log \frac{\phi_{uv}}{\theta_{uv}}$ . We define  $\mathcal{D}(\phi, \theta)$  to be the average KL-divergence over all histories:  $\mathcal{D}(\phi, \theta) \equiv \frac{1}{W} \sum_{u=1}^W KL(\phi_u \parallel \theta_u)$ , which is convex in  $\theta$  (Cover and Thomas, 1991). We will use the following derivative later:  $\partial \mathcal{D}(\phi, \theta) / \partial \theta_{uv} = -\phi_{uv} / (W \theta_{uv})$ .

We are now ready to define the regularized optimization problem for recovering a bigram LM  $\theta$  from the BOW corpus:

$$\begin{aligned} \max_{\theta} \quad & \ell(\theta) - \lambda \mathcal{D}(\phi, \theta) \\ \text{subject to} \quad & \theta \mathbf{1} = \mathbf{1}, \quad \theta \geq 0. \end{aligned} \quad (2)$$

The weight  $\lambda$  controls the strength of the prior. The constraints ensure that  $\theta$  is a valid bigram matrix, where  $\mathbf{1}$  is an all-one vector, and the non-negativity constraint is element-wise. Equivalently, (2) can be viewed as the *maximum a posteriori* (MAP) estimate of  $\theta$ , with independent Dirichlet priors for each row of  $\theta$ :  $p(\theta_u) = \text{Dir}(\theta_u | \alpha_u)$  and hyperparameters  $\alpha_{uv} = \frac{\lambda C}{W} \phi_{uv} + 1$ .

The summation over hidden ordered documents  $\mathbf{z}$  in  $P(\mathbf{x}|\theta)$  couples the variables and makes (2) a non-concave problem. We optimize  $\theta$  using an EM algorithm.

#### 3.2 The EM Algorithm

We derive the EM algorithm for the optimization problem (2). Let  $\mathcal{O}(\theta) \equiv \ell(\theta) - \lambda \mathcal{D}(\phi, \theta)$  be the objective function. Let  $\theta^{(t-1)}$  be the bigram LM at iteration  $t - 1$ . We can lower-bound  $\mathcal{O}$  as follows:

$$\begin{aligned} \mathcal{O}(\theta) &= \frac{1}{C} \sum_{i=1}^n \log \sum_{\mathbf{z} \in \sigma(\mathbf{x}_i)} P(\mathbf{z}|\theta^{(t-1)}, \mathbf{x}_i) \frac{P(\mathbf{z}|\theta)}{P(\mathbf{z}|\theta^{(t-1)}, \mathbf{x}_i)} \\ &\quad - \lambda \mathcal{D}(\phi, \theta) \\ &\geq \frac{1}{C} \sum_{i=1}^n \sum_{\mathbf{z} \in \sigma(\mathbf{x}_i)} P(\mathbf{z}|\theta^{(t-1)}, \mathbf{x}_i) \log \frac{P(\mathbf{z}|\theta)}{P(\mathbf{z}|\theta^{(t-1)}, \mathbf{x}_i)} \\ &\quad - \lambda \mathcal{D}(\phi, \theta) \\ &\equiv \mathcal{L}(\theta, \theta^{(t-1)}). \end{aligned}$$

We used Jensen’s inequality above since  $\log(\cdot)$  is concave. The lower bound  $\mathcal{L}$  involves  $P(\mathbf{z}|\theta^{(t-1)}, \mathbf{x})$ , the probability of hidden orderings of the BOW under the previous iteration’s model. In the E-step of EM we compute  $P(\mathbf{z}|\theta^{(t-1)}, \mathbf{x})$ , which will be discussed in Section 3.3. One can verify that  $\mathcal{L}(\theta, \theta^{(t-1)})$  is *concave* in  $\theta$ , unlike the original objective  $\mathcal{O}(\theta)$ . In addition, the lower bound ‘‘touches’’ the objective at  $\theta^{(t-1)}$ , i.e.,  $\mathcal{L}(\theta^{(t-1)}, \theta^{(t-1)}) = \mathcal{O}(\theta^{(t-1)})$ .

The EM algorithm iteratively maximizes the lower bound, which is now a concave optimization problem:  $\max_{\theta} \mathcal{L}(\theta, \theta^{(t-1)})$ , subject to  $\theta \mathbf{1} = \mathbf{1}$ . The non-negativity constraints turn out to be automatically satisfied. Introducing Lagrange multipliers  $\beta_u$  for each history  $u = 1 \dots W$ , we form the Lagrangian  $\Delta$ :

$$\Delta \equiv \mathcal{L}(\theta, \theta^{(t-1)}) - \sum_{u=1}^W \beta_u \left( \sum_{v=1}^W \theta_{uv} - 1 \right).$$

Taking the partial derivative with respect to  $\theta_{uv}$  and setting it to zero:  $\partial \Delta / \partial \theta_{uv} = 0$ , we arrive at the following update:

$$\theta_{uv} \propto \sum_{i=1}^n \sum_{\mathbf{z} \in \sigma(\mathbf{x}_i)} P(\mathbf{z}|\theta^{(t-1)}, \mathbf{x}_i) c_{uv}(\mathbf{z}) + \frac{\lambda C}{W} \phi_{uv}. \quad (3)$$

<p>Input: BOW documents <math>\{\mathbf{x}_1, \dots, \mathbf{x}_n\}</math>, a prior bigram LM <math>\phi</math>, weight <math>\lambda</math>.</p> <ol style="list-style-type: none"> <li>1. <math>t = 1</math>. Initialize <math>\theta^{(0)} = \phi</math>.</li> <li>2. Repeat until the objective <math>\mathcal{O}(\theta)</math> converges: <ol style="list-style-type: none"> <li>(a) (E-step) Compute <math>P(\mathbf{z} \theta^{(t-1)}, \mathbf{x})</math> for <math>\mathbf{z} \in \sigma(\mathbf{x}_i), i = 1, \dots, n</math>.</li> <li>(b) (M-step) Compute <math>\theta^{(t)}</math> using (3). Let <math>t = t + 1</math>.</li> </ol> </li> </ol> <p>Output: The recovered bigram LM <math>\theta</math>.</p>
--

Table 1: The EM algorithm

The normalization is over  $v = 1 \dots W$ . We use  $c_{uv}(\mathbf{z})$  to denote the number of times the bigram “ $uv$ ” appears in the ordered document  $\mathbf{z}$ . This is the M-step of EM. Intuitively, the first term counts how often the bigram “ $uv$ ” occurs, weighing each ordering by its probability under the previous model; the second term pulls the parameter towards the prior. If the weight of the prior  $\lambda \rightarrow \infty$ , we would have  $\theta_{uv} = \phi_{uv}$ . The update is related to the MAP estimate for a multinomial distribution with a Dirichlet prior, where we use the expected counts.

We initialize the EM algorithm with  $\theta^{(0)} = \phi$ . The EM algorithm is summarized in Table 1.

### 3.3 Approximate E-step

The E-step needs to compute the expected bigram counts of the form

$$\sum_{\mathbf{z} \in \sigma(\mathbf{x})} P(\mathbf{z}|\theta, \mathbf{x}) c_{uv}(\mathbf{z}). \quad (4)$$

However, this poses a computational problem. The summation is over unique ordered documents. The number of unique ordered documents can be on the order of  $|\mathbf{x}|!$ , i.e., all permutations of the BOW. For a short document of length 15, this number is already  $10^{12}$ . Clearly, brute-force enumeration is only feasible for very short documents. Approximation is necessary to handle longer ones.

A simple Monte Carlo approximation to (4) would involve sampling ordered documents  $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L$  according to  $\mathbf{z}_i \sim P(\mathbf{z}|\theta, \mathbf{x})$ , and replacing (4) with  $\sum_{i=1}^L c_{uv}(\mathbf{z}_i)/L$ . This estimate is unbiased, and the variance decreases linearly

with the number of samples,  $L$ . However, sampling directly from  $P$  is difficult.

Instead, we sample ordered documents  $\mathbf{z}_i \sim R(\mathbf{z}_i|\theta, \mathbf{x})$  from a distribution  $R$  which is easy to generate, and construct an approximation using *importance sampling* (see, e.g., (Liu, 2001)). With each sample,  $\mathbf{z}_i$ , we associate a weight  $w_i \propto P(\mathbf{z}_i|\theta, \mathbf{x})/R(\mathbf{z}_i|\theta, \mathbf{x})$ . The importance sampling approximation to (4) is then given by  $(\sum_{i=1}^L w_i c_{uv}(\mathbf{z}_i))/(\sum_{i=1}^L w_i)$ . Re-weighting the samples in this fashion accounts for the fact that we are using a sampling distribution  $R$  which is different from the target distribution  $P$ , and guarantees that our approximation is asymptotically unbiased.

The quality of an importance sampling approximation is closely related to how closely  $R$  resembles  $P$ ; the more similar they are, the better the approximation, in general. Given a BOW  $\mathbf{x}$  and our current bigram model estimate,  $\theta$ , we generate one sample (an ordered document  $\mathbf{z}_i$ ) by sequentially drawing words from the bag, with probabilities proportional to  $\theta$ , but properly normalized to form a distribution based on which words remain in the bag. For example, suppose  $\mathbf{x} = (\langle d \rangle:1, A:2, B:1, C:1)$ . Then we set  $z_{i1} = \langle d \rangle$ , and sample  $z_{i2} = A$  with probability  $2\theta_{\langle d \rangle A}/(2\theta_{\langle d \rangle A} + \theta_{\langle d \rangle B} + \theta_{\langle d \rangle C})$ . Similarly, if  $z_{i(j-1)} = u$  and if  $v$  is in the original BOW that hasn’t been sampled yet, then we set the next word in the ordered document  $z_{ij}$  equal to  $v$  with probability proportional to  $c_v \theta_{uv}$ , where  $c_v$  is the count of  $v$  in the remaining BOW. For this scheme, one can verify (Rabbat et al., 2007) that the importance weight corresponding to a sampled ordered document  $\mathbf{z}_i = (z_{i1}, \dots, z_{i|\mathbf{x}|})$  is given by  $w_i = \prod_{t=2}^{|\mathbf{x}|} \sum_{i=t}^{|\mathbf{x}|} \theta_{z_{t-1} z_i}$ . In our implementation, the number of importance samples used for a document  $\mathbf{x}$  is  $10|\mathbf{x}|^2$  if the length of the document  $|\mathbf{x}| > 8$ ; otherwise we enumerate  $\sigma(\mathbf{x})$  without importance sampling.

### 3.4 Prior Bigram LM $\phi$

The quality of the EM solution  $\theta$  can depend on the prior bigram LM  $\phi$ . To assess bigram recoverability from a BOW corpus alone, we consider only priors estimated from the corpus itself<sup>3</sup>. Like  $\theta$ ,  $\phi$  is a  $W \times W$  transition matrix with  $\phi_{uv} = P(v|u)$ . When

<sup>3</sup>Priors based on general English text or domain-specific knowledge could be used in specific applications.

appropriate, we set the initial probability  $\phi_{\langle d \rangle v}$  proportional to the number of times word  $v$  appears in the BOW corpus. We consider three prior models:

**Prior 1: Unigram  $\phi^{unigram}$ .** The most naïve  $\phi$  is a unigram LM which ignores word history. The probability for word  $v$  is estimated from the BOW corpus frequency of  $v$ , with add-1 smoothing:  $\phi_{uv}^{unigram} \propto 1 + \sum_{i=1}^n x_{iv}$ . We should point out that the unigram prior is an asymmetric bigram, i.e.,  $\phi_{uv}^{unigram} \neq \phi_{vu}^{unigram}$ .

**Prior 2: Frequency of Document Co-occurrence (FDC)  $\phi^{fdc}$ .** Let  $\delta(u, v | \mathbf{x}) = 1$  if words  $u \neq v$  co-occur (regardless of their counts) in BOW  $\mathbf{x}$ , and 0 otherwise. In the case  $u = v$ ,  $\delta(u, u | \mathbf{x}) = 1$  only if  $u$  appears at least twice in  $\mathbf{x}$ . Let  $c_{uv}^{fdc} = \sum_{i=1}^n \delta(u, v | \mathbf{x}_i)$  be the number of BOWs in which  $u, v$  co-occur. The FDC prior is  $\phi_{uv}^{fdc} \propto c_{uv}^{fdc} + 1$ . The co-occurrence counts  $c^{fdc}$  are symmetric, but  $\phi^{fdc}$  is asymmetric because of normalization. FDC captures some notion of potential transitions from  $u$  to  $v$ . FDC is in spirit similar to Kneser-Ney smoothing (Kneser and Ney, 1995) and other methods that accumulate indicators of document membership.

**Prior 3: Permutation-Based (Perm)  $\phi^{perm}$ .** Recall that  $c_{uv}(\mathbf{z})$  is the number of times the bigram “ $uv$ ” appears in an ordered document  $\mathbf{z}$ . We define  $c_{uv}^{perm} = \sum_{i=1}^n \mathbb{E}_{\mathbf{z} \in \sigma(\mathbf{x}_i)} [c_{uv}(\mathbf{z})]$ , where the expectation is with respect to all unique orderings of each BOW. We make the zero-knowledge assumption of uniform probability over these orderings, rather than  $P(\mathbf{z} | \theta)$  as in the EM algorithm described above. EM will refine these estimates, though, so this is a natural starting point. Space precludes a full discussion, but it can be proven that  $c_{uv}^{perm} = \sum_{i=1}^n x_{iu}x_{iv} / |\mathbf{x}_i|$  if  $u \neq v$ , and  $c_{uu}^{perm} = \sum_{i=1}^n x_{iu}(x_{iu} - 1) / |\mathbf{x}_i|$ . Finally,  $\phi_{uv}^{perm} \propto c_{uv}^{perm} + 1$ .

### 3.5 Decoding Ordered Documents from BOWs

Given a BOW  $\mathbf{x}$  and a bigram LM  $\theta$ , we formulate document recovery as the problem  $\mathbf{z}^* = \operatorname{argmax}_{\mathbf{z} \in \sigma(\mathbf{x})} P(\mathbf{z} | \theta)$ . In fact, we can generate the top  $N$  candidate ordered documents in terms of  $P(\mathbf{z} | \theta)$ . We use A\* search to construct such an N-best list (Russell and Norvig, 2003). Each state is an ordered, partial document. Its successor states append one more unused word in  $\mathbf{x}$  to

the partial document. The actual cost  $g$  from the start (empty document) to a state is the log probability of the partial document under bigram  $\theta$ . We design a heuristic cost  $h$  from the state to the goal (complete document) that is *admissible*: the idea is to over-use the best bigram history for the remaining words in  $\mathbf{x}$ . Let the partial document end with word  $w_e$ . Let the count vector for the remaining BOW be  $(c_1, \dots, c_W)$ . One admissible heuristic is  $h = \log \prod_{u=1}^W P(u | bh(u); \theta)^{c_u}$ , where the “best history” for word type  $u$  is  $bh(u) = \operatorname{argmax}_v \theta_{vu}$ , and  $v$  ranges over the word types with non-zero counts in  $(c_1, \dots, c_W)$ , plus  $w_e$ . It is easy to see that  $h$  is an upper bound on the bigram log probability that the remaining words in  $\mathbf{x}$  can achieve.

We use a memory-bounded A\* search similar to (Russell, 1992), because long BOWs would otherwise quickly exhaust memory. When the priority queue grows larger than the bound, the worst states (in terms of  $g + h$ ) in the queue are purged. This necessitates a double-ended priority queue that can pop either the maximum or minimum item. We use an efficient implementation with Splay trees (Chong and Sahni, 2000). We continue running A\* after popping the goal state from its priority queue. Repeating this  $N$  times gives the N-best list.

## 4 Experiments

We show experimentally that the proposed algorithm is indeed able to recover reasonable bigram LMs from BOW corpora. We observe:

**1. Good test set perplexity:** Using test (held-out) set perplexity (PP) as an objective measure of LM quality, we demonstrate that our recovered bigram LMs are much better than naïve unigram LMs trained on the same BOW corpus. Furthermore, they are not far behind the “oracle” bigram LMs trained on *ordered documents* that correspond to the BOWs.

**2. Sensible bigram pairs:** We inspect the recovered bigram LMs and find that they assign higher probabilities to sensible bigram pairs (e.g., “i mean”, “oh boy”, “that’s funny”), and lower probabilities to nonsense pairs (e.g., “i yep”, “you let’s”, “right lot”).

**3. Document recovery from BOW:** With the bigram LMs, we show improved accuracy in recovering ordered documents from BOWs.

We describe these experiments in detail below.



Corpus	$ V $	# Docs	# Tokens	$ \bar{x} $
SV10	10	6775	7792	1.2
SV25	25	9778	13324	1.4
SV50	50	12442	20914	1.7
SV100	100	14602	28611	2.0
SV250	250	18933	51950	2.7
SV500	500	23669	89413	3.8
SumTime	882	3341	68815	20.6

Table 2: Corpora statistics: vocabulary size, document count, total token count, and mean document length.

#### 4.1 Corpora and Protocols

We note that although in principle our algorithm works on large corpora, the current implementation does not scale well (Table 3 last column). We therefore experimented on seven corpora with relatively small vocabulary sizes, and with short documents (mostly one sentence per document). Table 2 lists statistics describing the corpora. The first six contain text transcripts of conversational telephone speech from the small vocabulary “SVitchboard 1” data set. King et al. constructed each corpus from the full Switchboard corpus, with the restriction that the sentences use only words in the corresponding vocabulary (King et al., 2005). We refer to these corpora as SV10, SV25, SV50, SV100, SV250, and SV500. The seventh corpus comes from the SumTime-Meteo data set (Sripada et al., 2003), which contains real weather forecasts for offshore oil rigs in the North Sea. For the SumTime corpus, we performed sentence segmentation to produce documents, removed punctuation, and replaced numeric digits with a special token.

For each of the seven corpora, we perform 5-fold cross validation. We use four folds other than the  $k$ -th fold as the training set to train (recover) bigram LMs, and the  $k$ -th fold as the test set for evaluation. This is repeated for  $k = 1 \dots 5$ , and we report the average cross validation results. We distinguish the original *ordered documents* (training set  $\mathbf{z}_1, \dots, \mathbf{z}_n$ , test set  $\mathbf{z}_{n+1}, \dots, \mathbf{z}_m$ ) and the corresponding BOWs (training set  $\mathbf{x}_1 \dots \mathbf{x}_n$ , test set  $\mathbf{x}_{n+1} \dots \mathbf{x}_m$ ). In all experiments, we simply set the weight  $\lambda = 1$  in (2). Given a training set and a test set, we perform the following steps:

1. Build prior LMs  $\phi^X$  from the training BOW corpus  $\mathbf{x}_1, \dots, \mathbf{x}_n$ , for  $X = \text{unigram}, \text{fdc}, \text{perm}$ .
2. Recover the bigram LMs  $\theta^X$  with the EM al-

gorithm in Table 1, from the training BOW corpus  $\mathbf{x}_1, \dots, \mathbf{x}_n$  and using the prior from step 1.

3. Compute the MAP bigram LM from the *ordered* training documents  $\mathbf{z}_1, \dots, \mathbf{z}_n$ . We call this the “oracle” bigram LM because it uses order information (not available to our algorithm), and we use it as a lower-bound on perplexity.

4. Test all LMs on  $\mathbf{z}_{n+1}, \dots, \mathbf{z}_m$  by perplexity.

#### 4.2 Good Test Set Perplexity

Table 3 reports the 5-fold cross validation mean-test-set-PP values for all corpora, and the run time per EM iteration. Because of the long running time, we adopt the rule-of-thumb stopping criterion of “two EM iterations”. First, we observe that all bigram LMs perform better than unigram LMs  $\phi^{\text{unigram}}$  even though they are trained on the same BOW corpus. Second, all recovered bigram LMs  $\theta^X$  improved upon their corresponding baselines  $\phi^X$ . The difference across *every* row is statistically significant according to a two-tailed paired  $t$ -test with  $p < 0.05$ . The differences among  $\text{PP}(\theta^X)$  for the same corpus are also significant (except between  $\theta^{\text{unigram}}$  and  $\theta^{\text{perm}}$  for SV500). Finally, we observe that  $\theta^{\text{perm}}$  tends to be best for the smaller vocabulary corpora, whereas  $\theta^{\text{fdc}}$  dominates as the vocabulary grows.

To see how much better we could do if we had *ordered* training documents  $\mathbf{z}_1, \dots, \mathbf{z}_n$ , we present the mean-test-set-PP of “oracle” bigram LMs in Table 4. We used three smoothing methods to obtain oracle LMs: absolute discounting using a constant of 0.5 (we experimented with other values, but 0.5 worked best), Good-Turing, and interpolated Witten-Bell as implemented in the SRILM toolkit (Stolcke, 2002). We see that our recovered LMs (trained on *unordered* BOW documents), especially for small vocabulary corpora, are close to the oracles (trained on *ordered* documents). For the larger datasets, the recovery task is more difficult, and the gap between the oracle LMs and the  $\theta$  LMs widens. Note that the oracle LMs do much better than the recovered LMs on the SumTime corpus; we suspect the difference is due to the larger vocabulary and significantly higher average sentence length (see Table 2).

#### 4.3 Sensible Bigram Pairs

The next set of experiments compares the recovered bigram LMs to their corresponding prior LMs

Corpus	$X$	$PP(\phi^X)$	$PP(\theta^X)$	Time/ Iter
SV10	unigram	7.48	6.95	< 1s
	fdc	6.52	6.47	< 1s
	perm	6.50	6.45	< 1s
SV25	unigram	16.4	12.8	0.1s
	fdc	12.3	11.8	0.1s
	perm	12.2	11.7	0.1s
SV50	unigram	29.1	19.7	2s
	fdc	19.6	17.8	4s
	perm	19.5	17.7	5s
SV100	unigram	45.4	27.8	7s
	fdc	29.5	25.3	11s
	perm	30.0	25.6	11s
SV250	unigram	91.8	51.2	5m
	fdc	60.0	47.3	8m
	perm	65.4	49.7	8m
SV500	unigram	149.1	87.2	3h
	fdc	104.8	80.1	3h
	perm	123.9	87.4	3h
SumTime	unigram	129.7	81.8	4h
	fdc	103.2	77.7	4h
	perm	187.9	85.4	3h

Table 3: Mean test set perplexities of prior LMs and bigram LMs recovered after 2 EM iterations.

in terms of how they assign probabilities to word pairs. One naturally expects probabilities for frequently occurring bigrams to increase, while rare or nonsensical bigrams’ probabilities should decrease. For a prior-bigram pair  $(\phi, \theta)$ , we evaluate the change in probabilities by computing the ratio  $\rho_{hw} = \frac{P(w|h, \theta)}{P(w|h, \phi)} = \frac{\theta_{hw}}{\phi_{hw}}$ . For a given history  $h$ , we sort words  $w$  by this ratio rather than by actual bigram probability because the bigrams with the highest and lowest probabilities tend to stay the same, while the changes accounting for differences in PP scores are more noticeable by considering the ratio.

Due to space limitation, we present one specific result (FDC prior, fold 1) for the SV500 corpus in Table 5. Other results are similar. The table lists a few most frequent unigrams as history words  $h$  (left), and the words  $w$  with the smallest (center) and largest (right)  $\rho_{hw}$  ratio. Overall we see that our EM algorithm is forcing meaningless bigrams (e.g., “i goodness”, “oh thing”) to have lower probabilities, while assigning higher probabilities to sensible bigram pairs (e.g., “really good”, “that’s funny”). Note that the reverse of some common expressions (e.g., “right that’s”) also rise in probability, suggesting the algorithm detects that the two words are of-

Corpus	Absolute Discount	Good-Turing	Witten-Bell	$\theta^*$
SV10	6.27	6.28	6.27	6.45
SV25	10.5	10.6	10.5	11.7
SV50	14.8	14.9	14.8	17.7
SV100	20.0	20.1	20.0	25.3
SV250	33.7	33.7	33.8	47.3
SV500	50.9	50.9	51.3	80.1
SumTime	10.8	10.5	10.6	77.7

Table 4: Mean test set perplexities for oracle bigram LMs trained on  $\mathbf{z}_1, \dots, \mathbf{z}_n$  and tested on  $\mathbf{z}_{n+1}, \dots, \mathbf{z}_m$ . For reference, the rightmost column lists the best result using a recovered bigram LM ( $\theta^{perm}$  for the first three corpora,  $\theta^{fdc}$  for the latter four).

ten adjacent, but lacks sufficient information to nail down the exact order.

#### 4.4 Document Recovery from BOW

We now play the role of the malicious party mentioned in the introduction. We show that, compared to their corresponding prior LMs, our recovered bigram LMs are better able to reconstruct ordered documents out of test BOWs  $\mathbf{x}_{n+1}, \dots, \mathbf{x}_m$ . We perform document recovery using 1-best A\* decoding. We use “document accuracy” and “ $n$ -gram accuracy” (for  $n = 2, 3$ ) as our evaluation criteria. We define document accuracy ( $Acc^{doc}$ ) as the fraction of documents<sup>4</sup> for which the decoded document matches the true ordered document exactly. Similarly,  $n$ -gram accuracy ( $Acc^n$ ) measures the fraction of all  $n$ -grams in test documents (with  $n$  or more words) that are recovered correctly.

For this evaluation, we compare models built for the SV500 corpus. Table 6 presents 5-fold cross validation average test-set accuracies. For each accuracy measure, we compare the prior LM with the recovered bigram LM. It is interesting to note that the FDC and Perm priors reconstruct documents surprisingly well, but we can always improve them by running our EM algorithm. The accuracies obtained by  $\theta$  are statistically significantly better (via two-tailed paired  $t$ -tests with  $p < 0.05$ ) than their corresponding priors  $\phi$  in all cases except  $Acc^{doc}$  for  $\theta^{perm}$  versus  $\phi^{perm}$ . Furthermore,  $\theta^{fdc}$  and  $\theta^{perm}$  are significantly better than all other models in terms of all three reconstruction accuracy measures.

<sup>4</sup>We omit single-word documents from these computations.

$h$	$w$ (smallest $\rho_{hw}$ )	$w$ (largest $\rho_{hw}$ )
i	yep, bye-bye, ah, goodness, ahead	mean, guess, think, bet, agree
you	let's, us, fact, such, deal	thank, bet, know, can, do
right	as, lot, going, years, were	that's, all, right, now, you're
oh	thing, here, could, were, doing	boy, really, absolutely, gosh, great
that's	talking, home, haven't, than, care	funny, wonderful, true, interesting, amazing
really	now, more, yep, work, you're	sad, neat, not, good, it's

Table 5: The recovered bigram LM  $\theta^{fdc}$  decreases nonsense bigram probabilities (center column) and increases sensible ones (right column) compared to the prior  $\phi^{fdc}$  on the SV500 corpus.

$\phi^{perm}$ reconstructions of test BOWs	$\theta^{perm}$ reconstructions of test BOWs
just it's it's it's just going it's probably out there else something the the have but it doesn't you to talking nice was it yes that's well that's what i'm saying a little more here home take and they can very be nice too i think well that's great i'm but was he because only always that's think i don't i no that in and it it's interesting <b>that's right that's right that's difficult</b> <b>so just not quite a year</b> <b>well it is a big dog</b> <b>so do you have a car</b>	<b>it's just it's just it's going</b> <b>it's probably something else out there</b> <b>but it doesn't have the the</b> <b>yes it was nice talking to you</b> <b>well that's that's what i'm saying</b> <b>a little more take home here</b> <b>and they can be very nice too</b> <b>well i think that's great i'm</b> <b>but only because he was always</b> <b>no i don't i think that's</b> <b>and it it's interesting that in</b> right that's that's right that's difficult so just not a quite year well it is big a dog so you do have a car

Table 7: Subset of SV500 documents that only  $\phi^{perm}$  or  $\theta^{perm}$  (but not both) reconstructs correctly. The correct reconstructions are in bold.

$X$	$Acc^{doc}$		$Acc^2$		$Acc^3$	
	$\phi^X$	$\theta^X$	$\phi^X$	$\theta^X$	$\phi^X$	$\theta^X$
unigram	11.1	26.8	17.7	32.8	2.7	11.8
fdc	30.2	31.0	33.0	35.1	11.4	13.3
perm	30.9	31.5	32.7	34.8	11.5	13.1

Table 6: Percentage of correctly reconstructed documents, 2-grams and 3-grams from test BOWs in SV500, 5-fold cross validation. The same trends continue for 4-grams and 5-grams (not shown).

We conclude our experiments with a closer look at some BOWs for which  $\phi$  and  $\theta$  reconstruct differently. As a representative example, we compare  $\theta^{perm}$  to  $\phi^{perm}$  on one test set of the SV500 corpus. There are 92 documents that are correctly reconstructed by  $\theta^{perm}$  but not by  $\phi^{perm}$ . In contrast, only 65 documents are accurately reordered by  $\phi^{perm}$  but not by  $\theta^{perm}$ . Table 7 presents a subset of these documents with six or more words. Overall, we conclude that the recovered bigram LMs do a better job at reconstructing BOW documents.

## 5 Conclusions and Future Work

We presented an algorithm that learns bigram language models from BOWs. We plan to: i) investigate ways to speed up our algorithm; ii) extend it to trigram and higher-order models; iii) handle the mixture of BOW documents and some ordered documents (or phrases) when available; iv) adapt a general English LM to a special domain using only BOWs from that domain; and v) explore novel applications of our algorithm.

## Acknowledgments

We thank Ben Liblit for tips on doubled-ended priority queues, and the anonymous reviewers for valuable comments. This work is supported in part by the Wisconsin Alumni Research Foundation, NSF CCF-0353079 and CCF-0728767, and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

## References

- Samit Basu and Nigel Boston. 2000. Identifiability of polynomial systems. Technical report, University of Illinois at Urbana-Champaign.
- Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.
- Stanley F. Chen and Joshua T. Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–393.
- Kyun-Rak Chong and Sartaj Sahni. 2000. Correspondence-based data structures for double-ended priority queues. *The ACM Journal of Experimental Algorithmics*, 5(2).
- Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons, Inc.
- Simon King, Chris Bartels, and Jeff Bilmes. 2005. SVitchboard 1: Small vocabulary tasks from Switchboard 1. In *Interspeech 2005*, Lisbon, Portugal.
- Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *ICASSP*.
- Jun S. Liu. 2001. *Monte Carlo Strategies in Scientific Computing*. Springer.
- Michael Rabbat, Mário Figueiredo, and Robert Nowak. 2007. Inferring network structure from co-occurrences. In *Advances in Neural Information Processing Systems (NIPS) 20*.
- Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech and Language*, 21(2):373–392.
- Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8).
- Stuart Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, second edition.
- Stuart Russell. 1992. Efficient memory-bounded search methods. In *The 10th European Conference on Artificial Intelligence*.
- Somayajulu G. Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. 2003. Exploiting a parallel TEXT-DATA corpus. In *Proceedings of Corpus Linguistics*, pages 734–743, Lancaster, U.K.
- Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, Denver, Colorado.

# Semi-Supervised Sequential Labeling and Segmentation using Giga-word Scale Unlabeled Data

Jun Suzuki and Hideki Isozaki

NTT Communication Science Laboratories, NTT Corp.  
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan  
{jun, isozaki}@cslab.kecl.ntt.co.jp

## Abstract

This paper provides evidence that the use of more unlabeled data in semi-supervised learning can improve the performance of Natural Language Processing (NLP) tasks, such as part-of-speech tagging, syntactic chunking, and named entity recognition. We first propose a simple yet powerful semi-supervised discriminative model appropriate for handling large scale unlabeled data. Then, we describe experiments performed on widely used test collections, namely, PTB III data, CoNLL'00 and '03 shared task data for the above three NLP tasks, respectively. We incorporate up to 1G-words (one billion tokens) of unlabeled data, which is the largest amount of unlabeled data ever used for these tasks, to investigate the performance improvement. In addition, our results are superior to the best reported results for all of the above test collections.

## 1 Introduction

Today, we can easily find a large amount of unlabeled data for many supervised learning applications in Natural Language Processing (NLP). Therefore, to improve performance, the development of an effective framework for semi-supervised learning (SSL) that uses both labeled and unlabeled data is attractive for both the machine learning and NLP communities. We expect that such SSL will replace most supervised learning in real world applications.

In this paper, we focus on traditional and important NLP tasks, namely part-of-speech (POS) tagging, syntactic chunking, and named entity recognition (NER). These are also typical supervised learning applications in NLP, and are referred to as sequential labeling and segmentation problems. In some cases, these tasks have relatively large

amounts of labeled training data. In this situation, supervised learning can provide competitive results, and it is difficult to improve them any further by using SSL. In fact, few papers have succeeded in showing significantly better results than state-of-the-art supervised learning. Ando and Zhang (2005) reported a substantial performance improvement compared with state-of-the-art supervised learning results for syntactic chunking with the CoNLL'00 shared task data (Tjong Kim Sang and Buchholz, 2000) and NER with the CoNLL'03 shared task data (Tjong Kim Sang and Meulder, 2003).

One remaining question is the behavior of SSL when using as much labeled and unlabeled data as possible. This paper investigates this question, namely, the use of a large amount of unlabeled data in the presence of (fixed) large labeled data.

To achieve this, it is paramount to make the SSL method scalable with regard to the size of unlabeled data. We first propose a scalable model for SSL. Then, we apply our model to widely used test collections, namely Penn Treebank (PTB) III data (Marcus et al., 1994) for POS tagging, CoNLL'00 shared task data for syntactic chunking, and CoNLL'03 shared task data for NER. We used up to 1G-words (one billion tokens) of unlabeled data to explore the performance improvement with respect to the unlabeled data size. In addition, we investigate the performance improvement for 'unseen data' from the viewpoint of unlabeled data coverage. Finally, we compare our results with those provided by the best current systems.

The contributions of this paper are threefold. First, we present a simple, scalable, but powerful task-independent model for semi-supervised sequential labeling and segmentation. Second, we report the best current results for the widely used test

collections described above. Third, we confirm that the use of more unlabeled data in SSL can really lead to further improvements.

## 2 Conditional Model for SSL

We design our model for SSL as a natural semi-supervised extension of conventional supervised conditional random fields (CRFs) (Lafferty et al., 2001). As our approach for incorporating unlabeled data, we basically follow the idea proposed in (Suzuki et al., 2007).

### 2.1 Conventional Supervised CRFs

Let  $\mathbf{x} \in \mathcal{X}$  and  $\mathbf{y} \in \mathcal{Y}$  be an input and output, where  $\mathcal{X}$  and  $\mathcal{Y}$  represent the set of possible inputs and outputs, respectively.  $\mathcal{C}$  stands for the set of cliques in an undirected graphical model  $\mathcal{G}(\mathbf{x}, \mathbf{y})$ , which indicates the interdependency of a given  $\mathbf{x}$  and  $\mathbf{y}$ .  $\mathbf{y}_c$  denotes the output from the corresponding clique  $c$ . Each clique  $c \in \mathcal{C}$  has a *potential function*  $\Psi_c$ . Then, the CRFs define the conditional probability  $p(\mathbf{y}|\mathbf{x})$  as a product of  $\Psi_c$ s. In addition, let  $\mathbf{f} = (f_1, \dots, f_I)$  be a feature vector, and  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_I)$  be a parameter vector, whose lengths are  $I$ .  $p(\mathbf{y}|\mathbf{x}; \boldsymbol{\lambda})$  on a CRF is defined as follows:

$$p(\mathbf{y}|\mathbf{x}; \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x})} \prod_c \Psi_c(\mathbf{y}_c, \mathbf{x}; \boldsymbol{\lambda}), \quad (1)$$

where  $Z(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \prod_{c \in \mathcal{C}} \Psi_c(\mathbf{y}_c, \mathbf{x}; \boldsymbol{\lambda})$  is the partition function. We generally assume that the potential function is a non-negative real value function. Therefore, the exponentiated weighted sum over the features of a clique is widely used, so that,  $\Psi_c(\mathbf{y}_c, \mathbf{x}; \boldsymbol{\lambda}) = \exp(\boldsymbol{\lambda} \cdot \mathbf{f}_c(\mathbf{y}_c, \mathbf{x}))$  where  $\mathbf{f}_c(\mathbf{y}_c, \mathbf{x})$  is a feature vector obtained from the corresponding clique  $c$  in  $\mathcal{G}(\mathbf{x}, \mathbf{y})$ .

### 2.2 Semi-supervised Extension for CRFs

Suppose we have  $J$  kinds of probability models (PMs). The  $j$ -th joint PM is represented by  $p_j(\mathbf{x}_j, \mathbf{y}; \boldsymbol{\theta}_j)$  where  $\boldsymbol{\theta}_j$  is a model parameter.  $\mathbf{x}_j = \mathcal{T}_j(\mathbf{x})$  is simply an input  $\mathbf{x}$  transformed by a pre-defined function  $\mathcal{T}_j$ . We assume  $\mathbf{x}_j$  has the same graph structure as  $\mathbf{x}$ . This means  $p_j(\mathbf{x}_j, \mathbf{y})$  can be factorized by the cliques  $c$  in  $\mathcal{G}(\mathbf{x}, \mathbf{y})$ . That is,  $p_j(\mathbf{x}_j, \mathbf{y}; \boldsymbol{\theta}_j) = \prod_c p_j(\mathbf{x}_{jc}, \mathbf{y}_c; \boldsymbol{\theta}_j)$ . Thus, we can incorporate generative models such as Bayesian networks including (1D and 2D) hidden Markov models (HMMs) as these joint PMs. Actually, there is

a difference in that generative models are *directed* graphical models while our conditional PM is an *undirected*. However, this difference causes no violations when we construct our approach.

Let us introduce  $\boldsymbol{\lambda}' = (\lambda_1, \dots, \lambda_I, \lambda_{I+1}, \dots, \lambda_{I+J})$ , and  $\mathbf{h} = (f_1, \dots, f_I, \log p_1, \dots, \log p_J)$ , which is the concatenation of feature vector  $\mathbf{f}$  and the log-likelihood of  $J$ -joint PMs. Then, we can define a new potential function by embedding the joint PMs;

$$\begin{aligned} \Psi'_c(\mathbf{y}_c, \mathbf{x}; \boldsymbol{\lambda}', \boldsymbol{\Theta}) &= \exp(\boldsymbol{\lambda}' \cdot \mathbf{f}_c(\mathbf{y}_c, \mathbf{x})) \cdot \prod_j p_j(\mathbf{x}_{jc}, \mathbf{y}_c; \boldsymbol{\theta}_j)^{\lambda_{I+j}} \\ &= \exp(\boldsymbol{\lambda}' \cdot \mathbf{h}_c(\mathbf{y}_c, \mathbf{x})). \end{aligned}$$

where  $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_j\}_{j=1}^J$ , and  $\mathbf{h}_c(\mathbf{y}_c, \mathbf{x})$  is  $\mathbf{h}$  obtained from the corresponding clique  $c$  in  $\mathcal{G}(\mathbf{x}, \mathbf{y})$ . Since each  $p_j(\mathbf{x}_{jc}, \mathbf{y}_c)$  has range  $[0, 1]$ , which is non-negative,  $\Psi'_c$  can also be used as a potential function. Thus, the conditional model for our SSL can be written as:

$$P(\mathbf{y}|\mathbf{x}; \boldsymbol{\lambda}', \boldsymbol{\Theta}) = \frac{1}{Z'(\mathbf{x})} \prod_c \Psi'_c(\mathbf{y}_c, \mathbf{x}; \boldsymbol{\lambda}', \boldsymbol{\Theta}), \quad (2)$$

where  $Z'(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \prod_{c \in \mathcal{C}} \Psi'_c(\mathbf{y}_c, \mathbf{x}; \boldsymbol{\lambda}', \boldsymbol{\Theta})$ . Hereafter in this paper, we refer to this conditional model as a ‘*Joint probability model Embedding style Semi-Supervised Conditional Model*’, or **JESS-CM** for short.

Given labeled data,  $\mathcal{D}_l = \{(\mathbf{x}^n, \mathbf{y}^n)\}_{n=1}^N$ , the MAP estimation of  $\boldsymbol{\lambda}'$  under a fixed  $\boldsymbol{\Theta}$  can be written as:

$$\mathcal{L}^1(\boldsymbol{\lambda}'|\boldsymbol{\Theta}) = \sum_n \log P(\mathbf{y}^n|\mathbf{x}^n; \boldsymbol{\lambda}', \boldsymbol{\Theta}) + \log p(\boldsymbol{\lambda}'),$$

where  $p(\boldsymbol{\lambda}')$  is a prior probability distribution of  $\boldsymbol{\lambda}'$ . Clearly, JESS-CM shown in Equation 2 has exactly the same form as Equation 1. With a fixed  $\boldsymbol{\Theta}$ , the log-likelihood,  $\log p_j$ , can be seen simply as the feature functions of JESS-CM as with  $f_i$ . Therefore, embedded joint PMs do not violate the global convergence conditions. As a result, as with supervised CRFs, it is guaranteed that  $\boldsymbol{\lambda}'$  has a value that achieves the global maximum of  $\mathcal{L}^1(\boldsymbol{\lambda}'|\boldsymbol{\Theta})$ . Moreover, we can obtain the same form of gradient as that of supervised CRFs (Sha and Pereira, 2003), that is,

$$\begin{aligned} \nabla \mathcal{L}^1(\boldsymbol{\lambda}'|\boldsymbol{\Theta}) &= E_{\tilde{P}(\mathcal{Y}, \mathcal{X}; \boldsymbol{\lambda}', \boldsymbol{\Theta})}[\mathbf{h}(\mathcal{Y}, \mathcal{X})] \\ &\quad - \sum_n E_{P(\mathcal{Y}|\mathbf{x}^n; \boldsymbol{\lambda}', \boldsymbol{\Theta})}[\mathbf{h}(\mathcal{Y}, \mathbf{x}^n)] + \nabla \log p(\boldsymbol{\lambda}'). \end{aligned}$$

Thus, we can easily optimize  $\mathcal{L}^1$  by using the forward-backward algorithm since this paper solely

focuses on a sequence model and a gradient-based optimization algorithm in the same manner as those used in supervised CRF parameter estimation.

We cannot naturally incorporate unlabeled data into standard discriminative learning methods since the correct outputs  $\mathbf{y}$  for unlabeled data are unknown. On the other hand with a generative approach, a well-known way to achieve this incorporation is to use maximum marginal likelihood (MML) parameter estimation, i.e., (Nigam et al., 2000). Given unlabeled data  $\mathcal{D}_u = \{\mathbf{x}^m\}_{m=1}^M$ , MML estimation in our setting maximizes the marginal distribution of a joint PM over a missing (hidden) variable  $\mathbf{y}$ , namely, it maximizes  $\sum_m \log \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{x}^m, \mathbf{y}; \theta)$ .

Following this idea, there have been introduced a parameter estimation approach for non-generative approaches that can effectively incorporate unlabeled data (Suzuki et al., 2007). Here, we refer to it as ‘Maximum Discriminant Functions sum’ (MDF) parameter estimation. MDF estimation substitutes  $p(\mathbf{x}, \mathbf{y})$  with discriminant functions  $g(\mathbf{x}, \mathbf{y})$ . Therefore, to estimate the parameter  $\Theta$  of JESS-CM by using MDF estimation, the following objective function is maximized with a fixed  $\lambda'$ :

$$\mathcal{L}^2(\Theta|\lambda') = \sum_m \log \sum_{\mathbf{y} \in \mathcal{Y}} g(\mathbf{x}^m, \mathbf{y}; \lambda', \Theta) + \log p(\Theta),$$

where  $p(\Theta)$  is a prior probability distribution of  $\Theta$ . Since the normalization factor does not affect the determination of  $\mathbf{y}$ , the discriminant function of JESS-CM shown in Equation 2 is defined as  $g(\mathbf{x}, \mathbf{y}; \lambda', \Theta) = \prod_{c \in \mathcal{C}} \Psi'_c(\mathbf{y}_c, \mathbf{x}; \lambda', \Theta)$ . With a fixed  $\lambda'$ , the local maximum of  $\mathcal{L}^2(\Theta|\lambda')$  around the initialized value of  $\Theta$  can be estimated by an iterative computation such as the EM algorithm (Dempster et al., 1977).

### 2.3 Scalability: Efficient Training Algorithm

A parameter estimation algorithm of  $\lambda'$  and  $\Theta$  can be obtained by maximizing the objective functions  $\mathcal{L}^1(\lambda'|\Theta)$  and  $\mathcal{L}^2(\Theta|\lambda')$  iteratively and alternately. Figure 1 summarizes an algorithm for estimating  $\lambda'$  and  $\Theta$  for JESS-CM.

This paper considers a situation where there are many more unlabeled data  $M$  than labeled data  $N$ , that is,  $N \ll M$ . This means that the calculation cost for unlabeled data is dominant. Thus, in order to make the overall parameter estimation procedure

---

**Input:** training data  $\mathcal{D} = \{\mathcal{D}_l, \mathcal{D}_u\}$   
 where labeled data  $\mathcal{D}_l = \{(\mathbf{x}^n, \mathbf{y}^n)\}_{n=1}^N$ ,  
 and unlabeled data  $\mathcal{D}_u = \{\mathbf{x}^m\}_{m=1}^M$

**Initialize:**  $\Theta^{(0)} \leftarrow$  uniform distribution,  $t \leftarrow 0$

**do**

1.  $t \leftarrow t + 1$
2. (Re)estimate  $\lambda'$ :  
 maximize  $\mathcal{L}^1(\lambda'|\Theta)$  with fixed  $\Theta \leftarrow \Theta^{(t-1)}$  using  $\mathcal{D}_l$ .
3. Estimate  $\Theta^{(t)}$ : (Initial values =  $\Theta^{(t-1)}$ )  
 update one step toward maximizing  $\mathcal{L}^2(\Theta|\lambda')$   
 with fixed  $\lambda'$  using  $\mathcal{D}_u$ .

**do.until**  $\frac{|\Theta^{(t)} - \Theta^{(t-1)}|}{|\Theta^{(t-1)}|} < \epsilon$ .

Reestimate  $\lambda'$ : perform the same procedure as 1.

**Output:** a JESS-CM,  $P(\mathbf{y}|\mathbf{x}, \lambda', \Theta^{(t)})$ .

---

Figure 1: Parameter estimation algorithm for JESS-CM.

scalable for handling large scale unlabeled data, we only perform one step of MDF estimation for each  $t$  as explained on 3. in Figure 1. In addition, the calculation cost for estimating parameters of embedded joint PMs (HMMs) is independent of the number of HMMs,  $J$ , that we used (Suzuki et al., 2007). As a result, the cost for calculating the JESS-CM parameters,  $\lambda'$  and  $\Theta$ , is essentially the same as executing  $T$  iterations of the MML estimation for a single HMM using the EM algorithm plus  $T + 1$  time optimizations of the MAP estimation for a conventional supervised CRF if it converged when  $t = T$ . In addition, our parameter estimation algorithm can be easily performed in parallel computation.

### 2.4 Comparison with Hybrid Model

SSL based on a hybrid generative/discriminative approach proposed in (Suzuki et al., 2007) has been defined as a log-linear model that discriminatively combines several discriminative models,  $p_i^D$ , and generative models,  $p_j^G$ , such that:

$$R(\mathbf{y}|\mathbf{x}; \Lambda, \Theta, \Gamma) = \frac{\prod_i p_i^D(\mathbf{y}|\mathbf{x}; \lambda_i)^{\gamma_i} \prod_j p_j^G(\mathbf{x}_j, \mathbf{y}; \theta_j)^{\gamma_j}}{\sum_{\mathbf{y}} \prod_i p_i^D(\mathbf{y}|\mathbf{x}; \lambda_i)^{\gamma_i} \prod_j p_j^G(\mathbf{x}_j, \mathbf{y}; \theta_j)^{\gamma_j}},$$

where  $\Lambda = \{\lambda_i\}_{i=1}^I$ , and  $\Gamma = \{\{\gamma_i\}_{i=1}^I, \{\gamma_j\}_{j=I+1}^{I+J}\}$ .

With the hybrid model, if we use the same labeled training data to estimate both  $\Lambda$  and  $\Gamma$ ,  $\gamma_j$ s will become negligible (zero or nearly zero) since  $p_i^D$  is already fitted to the labeled training data while  $p_j^G$  are trained by using unlabeled data. As a solution, a given amount of labeled training data is divided into two distinct sets, i.e., 4/5 for estimating  $\Lambda$ , and the

remaining 1/5 for estimating  $\Gamma$  (Suzuki et al., 2007). Moreover, it is necessary to split features into several sets, and then train several corresponding discriminative models separately and preliminarily. In contrast, JESS-CM is free from this kind of additional process, and the entire parameter estimation procedure can be performed in a single pass. Surprisingly, although JESS-CM is a simpler version of the hybrid model in terms of model structure and parameter estimation procedure, JESS-CM provides  $F$ -scores of 94.45 and 88.03 for CoNLL’00 and ’03 data, respectively, which are 0.15 and 0.83 points higher than those reported in (Suzuki et al., 2007) for the same configurations. This performance improvement is basically derived from the full benefit of using labeled training data for estimating the parameter of the conditional model while the combination weights,  $\Gamma$ , of the hybrid model are estimated solely by using 1/5 of the labeled training data. These facts indicate that JESS-CM has several advantageous characteristics compared with the hybrid model.

### 3 Experiments

In our experiments, we report POS tagging, syntactic chunking and NER performance incorporating up to 1G-words of unlabeled data.

#### 3.1 Data Set

To compare the performance with that of previous studies, we selected widely used test collections. For our POS tagging experiments, we used the Wall Street Journal in PTB III (Marcus et al., 1994) with the same data split as used in (Shen et al., 2007). For our syntactic chunking and NER experiments, we used exactly the same training, development and test data as those provided for the shared tasks of CoNLL’00 (Tjong Kim Sang and Buchholz, 2000) and CoNLL’03 (Tjong Kim Sang and Meulder, 2003), respectively. The training, development and test data are detailed in Table 1<sup>1</sup>.

The unlabeled data for our experiments was taken from the Reuters corpus, TIPSTER corpus (LDC93T3C) and the English Gigaword corpus, third edition (LDC2007T07). As regards the TIP-

<sup>1</sup>The second-order encoding used in our NER experiments is the same as that described in (Sha and Pereira, 2003) except removing IOB-tag of previous position label.

(a) POS-tagging: (WSJ in PTB III)			
# of labels			45
Data set	(WSJ sec. IDs)	# of sent.	# of words
Training	0–18	38,219	912,344
Development	19–21	5,527	131,768
Test	22–24	5,462	129,654

(b) Chunking: (WSJ in PTB III: CoNLL’00 shared task data)			
# of labels			23 (w/ IOB-tagging)
Data set	(WSJ sec. IDs)	# of sent.	# of words
Training	15–18	8,936	211,727
Development	N/A	N/A	N/A
Test	20	2,012	47,377

(c) NER: (Reuters Corpus: CoNLL’03 shared task data)			
# of labels			29 (w/ IOB-tagging+2nd-order encoding)
Data set	(time period)	# of sent.	# of words
Training	22–30/08/96	14,987	203,621
Development	30–31/08/96	3,466	51,362
Test	06–07/12/96	3,684	46,435

Table 1: Details of training, development, and test data (labeled data set) used in our experiments

data	abbr.	(time period)	# of sent.	# of words
Tipster	wsj	04/90–03/92	1,624,744	36,725,301
Reuters Corpus	reu	09/96–08/97* *(excluding 06–07/12/96)	13,747,227	215,510,564
English Gigaword	afp	05/94–12/96	5,510,730	135,041,450
	apw	11/94–12/96	7,207,790	154,024,679
	ltw	04/94–12/96	3,094,290	72,928,537
	nyt	07/94–12/96	15,977,991	357,952,297
	xin	01/95–12/96	1,740,832	40,078,312
total	all		48,903,604	1,012,261,140

Table 2: Unlabeled data used in our experiments

STER corpus, we extracted all the Wall Street Journal articles published between 1990 and 1992. With the English Gigaword corpus, we extracted articles from five news sources published between 1994 and 1996. The unlabeled data used in this paper is detailed in Table 2. Note that the total size of the unlabeled data reaches 1G-words (one billion tokens).

#### 3.2 Design of JESS-CM

We used the same graph structure as the linear chain CRF for JESS-CM. As regards the design of the feature functions  $f_i$ , Table 3 shows the feature templates used in our experiments. In the table,  $s$  indicates a focused token position.  $X_{s-1:s}$  represents the bi-gram of feature  $X$  obtained from  $s-1$  and  $s$  positions.  $\{X_u\}_{u=A}^B$  indicates that  $u$  ranges from  $A$  to  $B$ . For example,  $\{X_u\}_{u=s-2}^{s+2}$  is equal to five feature templates,  $\{X_{s-2}, X_{s-1}, X_s, X_{s+1}, X_{s+2}\}$ . ‘word type’ or wtp represents features of a word such as capitalization, the existence of digits, and punctuation as shown in (Sutton et al., 2006) without regular expressions. Although it is common to use external



(a) POS tagging:(total 47 templates)
$[y_s], [y_{s-1:s}], \{[y_s, \text{pf-N}_s], [y_s, \text{sf-N}_s]\}_{N=1}^9,$ $\{[y_s, \text{wd}_u], [y_s, \text{wtp}_u], [y_{s-1:s}, \text{wtp}_u]\}_{u=s-2}^{s+2},$ $\{[y_s, \text{wd}_{u-1:u}], [y_s, \text{wtp}_{u-1:u}], [y_{s-1:s}, \text{wtp}_{u-1:u}]\}_{u=s-1}^{s+2}$
(b) Syntactic chunking: (total 39 templates)
$[y_s], [y_{s-1:s}], \{[y_s, \text{wd}_u], [y_s, \text{pos}_u], [y_s, \text{wd}_u, \text{pos}_u],$ $[y_{s-1:s}, \text{wd}_u], [y_{s-1:s}, \text{pos}_u]\}_{u=s-2}^{s+2}, \{[y_s, \text{wd}_{u-1:u}],$ $[y_s, \text{pos}_{u-1:u}], \{[y_{s-1:s}, \text{pos}_{u-1:u}]\}_{u=s-1}^{s+2},$
(c) NER: (total 79 templates)
$[y_s], [y_{s-1:s}], \{[y_s, \text{wd}_u], [y_s, \text{lwd}_u], [y_s, \text{pos}_u], [y_s, \text{wtp}_u],$ $[y_{s-1:s}, \text{lwd}_u], [y_{s-1:s}, \text{pos}_u], [y_{s-1:s}, \text{wtp}_u]\}_{u=s-2}^{s+2},$ $\{[y_s, \text{lwd}_{u-1:u}], [y_s, \text{pos}_{u-1:u}], [y_s, \text{wtp}_{u-1:u}],$ $[y_{s-1:s}, \text{pos}_{u-1:u}], [y_{s-1:s}, \text{wtp}_{u-1:u}]\}_{u=s-1}^{s+2},$ $[y_s, \text{pos}_{s-1:s+1}], [y_s, \text{wtp}_{s-1:s+1}], [y_{s-1:s}, \text{pos}_{s-1:s+1}],$ $[y_{s-1:s}, \text{wtp}_{s-1:s+1}], [y_s, \text{wd4l}_s], [y_s, \text{wd4r}_s],$ $\{[y_s, \text{pf-N}_s], [y_s, \text{sf-N}_s], [y_{s-1:s}, \text{pf-N}_s], [y_{s-1:s}, \text{sf-N}_s]\}_{N=1}^4$
wd: word, pos: part-of-speech lwd: lowercase of word, wtp: 'word type', wd4{l,r}: words within the left or right 4 tokens {pf,sf}-N: N character prefix or suffix of word

Table 3: Feature templates used in our experiments

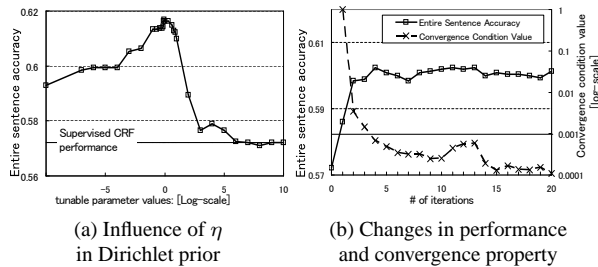


Figure 2: Typical behavior of tunable parameters

resources such as gazetteers for NER, we used none. All our features can be automatically extracted from the given training data.

### 3.3 Design of Joint PMs (HMMs)

We used first order HMMs for embedded joint PMs since we assume that they have the same graph structure as JESS-CM as described in Section 2.2.

To reduce the required human effort, we simply used the feature templates shown in Table 3 to generate the features of the HMMs. With our design, one feature template corresponded to one HMM. This design preserves the feature whereby each HMM emits a single symbol from a single state (or transition). We can easily ignore overlapping features that appear in a single HMM. As a result, 47, 39 and 79 distinct HMMs are embedded in the potential functions of JESS-CM for POS tagging, chunking and NER experiments, respectively.

### 3.4 Tunable Parameters

In our experiments, we selected Gaussian and Dirichlet priors as the prior distributions in  $\mathcal{L}^1$  and

$\mathcal{L}^2$ , respectively. This means that JESS-CM has two tunable parameters,  $\sigma^2$  and  $\eta$ , in the Gaussian and Dirichlet priors, respectively. The values of these tunable parameters are chosen by employing a binary line search. We used the value for the best performance with the development set<sup>2</sup>. However, it may be computationally unrealistic to retrain the entire procedure several times using 1G-words of unlabeled data. Therefore, these tunable parameter values are selected using a relatively small amount of unlabeled data (17M-words), and we used the selected values in all our experiments. The left graph in Figure 2 shows typical  $\eta$  behavior. The left end is equivalent to optimizing  $\mathcal{L}^2$  without a prior, and the right end is almost equivalent to considering  $p_j(\mathbf{x}_j, \mathbf{y})$  for all  $j$  to be a uniform distribution. This is why it appears to be bounded by the performance obtained from supervised CRF. We omitted the influence of  $\sigma^2$  because of space constraints, but its behavior is nearly the same as that of supervised CRF.

Unfortunately,  $\mathcal{L}^2(\Theta|\lambda')$  may have two or more local maxima. Our parameter estimation procedure does not guarantee to provide either the global optimum or a convergence solution in  $\Theta$  and  $\lambda'$  space. An example of non-convergence is the oscillation of the estimated  $\Theta$ . That is,  $\Theta$  traverses two or more local maxima. Therefore, we examined its convergence property experimentally. The right graph in Figure 2 shows a typical convergence property. Fortunately, in all our experiments, JESS-CM converged in a small number of iterations. No oscillation is observed here.

## 4 Results and Discussion

### 4.1 Impact of Unlabeled Data Size

Table 4 shows the performance of JESS-CM using 1G-words of unlabeled data and the performance gain compared with supervised CRF, which is trained under the same conditions as JESS-CM except that joint PMs are not incorporated. We emphasize that our model achieved these large improvements solely using unlabeled data as additional resources, without introducing a sophisticated model, deep feature engineering, handling external hand-

<sup>2</sup>Since CoNLL'00 shared task data has no development set, we divided the labeled training data into two distinct sets, 4/5 for training and the remainder for the development set, and determined the tunable parameters in preliminary experiments.

	(a) POS tagging				(b) Chunking		(c) NER			
measures	label accuracy		entire sent. acc.		$F_{\beta=1}$	sent. acc.	$F_{\beta=1}$		entire sent. acc.	
eval. data	dev.	test	dev.	test	test	test	dev.	test	dev.	test
<b>JESS-CM (CRF/HMM)</b>	97.35	97.40	56.34	57.01	95.15	65.06	94.48	89.92	91.17	85.12
(gain from supervised CRF)	(+0.17)	(+0.19)	(+1.90)	(+1.63)	(+1.27)	(+4.92)	(+2.74)	(+3.57)	(+3.46)	(+3.96)

Table 4: Results for POS tagging (PTB III data), syntactic chunking (CoNLL’00 data), and NER (CoNLL’03 data) incorporated with 1G-words of unlabeled data, and the performance gain from supervised CRF

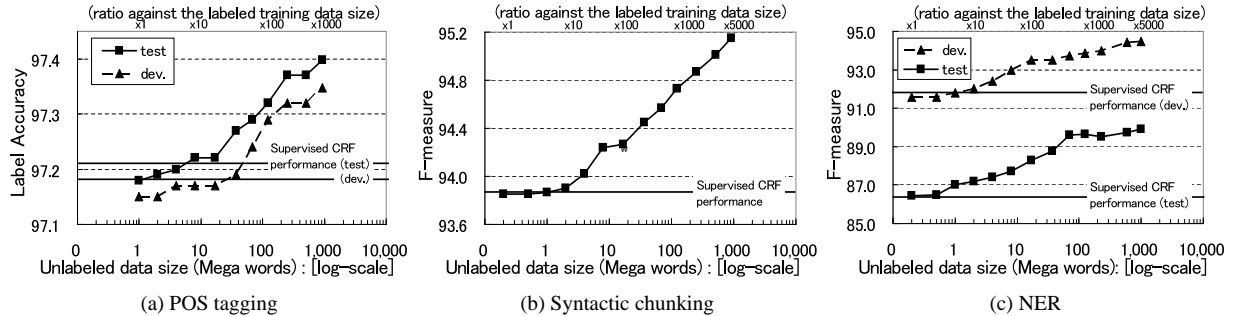


Figure 3: Performance changes with respect to unlabeled data size in JESS-CM

crafted resources, or task dependent human knowledge (except for the feature design). Our method can greatly reduce the human effort needed to obtain a high performance tagger or chunker.

Figure 3 shows the learning curves of JESS-CM with respect to the size of the unlabeled data, where the x-axis is on the logarithmic scale of the unlabeled data size (Mega-word). The scale at the top of the graph shows the ratio of the unlabeled data size to the labeled data size. We observe that a small amount of unlabeled data hardly improved the performance since the supervised CRF results are competitive. It seems that we require at least dozens of times more unlabeled data than labeled training data to provide a significant performance improvement. The most important and interesting behavior is that the performance improvements against the unlabeled data size are almost linear on a logarithmic scale within the size of the unlabeled data used in our experiments. Moreover, there is a possibility that the performance is still unsaturated at the 1G-word unlabeled data point. This suggests that increasing the unlabeled data in JESS-CM may further improve the performance.

Suppose  $J=1$ , the discriminant function of JESS-CM is  $g(x, \mathbf{y}) = \mathcal{A}(x, \mathbf{y})p_1(x_1, \mathbf{y}; \theta_1)^{\lambda_{I+1}}$  where  $\mathcal{A}(x, \mathbf{y}) = \exp(\lambda \cdot \sum_c \mathbf{f}_c(\mathbf{y}_c, x))$ . Note that both  $\mathcal{A}(x, \mathbf{y})$  and  $\lambda_{I+j}$  are given and fixed during the MDF estimation of joint PM parameters  $\Theta$ . Therefore, the MDF estimation in JESS-CM can be re-

garded as a variant of the MML estimation (see Section 2.2), namely, it is MML estimation with a bias,  $\mathcal{A}(x, \mathbf{y})$ , and smooth factors,  $\lambda_{I+j}$ . MML estimation can be seen as modeling  $p(x)$  since it is equivalent to maximizing  $\sum_m \log p(x^m)$  with marginalized hidden variables  $\mathbf{y}$ , where  $\sum_{\mathbf{y} \in \mathcal{Y}} p(x, \mathbf{y}) = p(x)$ . Generally, more data will lead to a more accurate model of  $p(x)$ . With our method, as with modeling  $p(x)$  in MML estimation, more unlabeled data is preferable since it may provide more accurate modeling. This also means that it provides better ‘clusters’ over the output space since  $\mathcal{Y}$  is used as hidden states in HMMs. These are intuitive explanations as to why more unlabeled data in JESS-CM produces better performance.

## 4.2 Expected Performance for Unseen Data

We try to investigate the impact of unlabeled data on the performance of unseen data. We divide the test set (or the development set) into two disjoint sets: L.app and L.neg app. **L.app** is a set of sentences constructed by words that all **appeared** in the Labeled training data. **L.-app** is a set of sentences that have at least one word that does **not appear** in the Labeled training data.

Table 5 shows the performance with these two sets obtained from both supervised CRF and JESS-CM with 1G-word unlabeled data. As the supervised CRF results, the performance of the L.-app sets is consistently much lower than that of the cor-

eval. data	(a) POS tagging				(b) Chunking		(c) NER			
	development		test		test		development		test	
	L. $\neg$ app (46.1%)	L.app (53.9%)	L. $\neg$ app (40.4%)	L.app (59.6%)	L. $\neg$ app (70.7%)	L.app (29.3%)	L. $\neg$ app (54.3%)	L.app (45.7%)	L. $\neg$ app (64.3%)	L.app (35.7%)
supervised CRF (baseline)	<b>46.78</b>	60.99	<b>48.57</b>	60.01	<b>56.92</b>	67.91	<b>79.60</b>	97.35	<b>75.69</b>	91.03
<b>JESS-CM</b> (CRF/HMM)	49.02	62.60	50.79	61.24	62.47	71.30	85.87	97.47	80.84	92.85
(gain from supervised CRF)	<b>(+2.24)</b>	<b>(+1.61)</b>	<b>(+2.22)</b>	<b>(+1.23)</b>	<b>(+5.55)</b>	<b>(+3.40)</b>	<b>(+6.27)</b>	<b>(+0.12)</b>	<b>(+5.15)</b>	<b>(+1.82)</b>
U.app	83.7%	96.3%	84.3%	95.8%	89.5%	99.2%	95.3%	99.8%	94.9%	100.0%

Table 5: Comparison with L. $\neg$ app and L.app sets obtained from both supervised CRF and JESS-CM with 1G-word unlabeled data evaluated by the **entire sentence accuracies**, and the ratio of U.app.

unlab. data		dev (Aug. 30-31)		test (Dec. 06-07)		
(period)	#sent.	#wds	$F_{\beta=1}$	U.app	$F_{\beta=1}$	U.app
reu(Sep.)	1.0M	17M	93.50	82.0%	88.27	69.7%
reu(Oct.)	1.3M	20M	93.04	71.0%	88.82	72.0%
reu(Nov.)	1.2M	18M	92.94	68.7%	89.08	74.3%
reu(Dec.)*	9M	15M	92.91	67.0%	89.29	84.4%

Table 6: Influence of U.app in NER experiments: \*(excluding Dec. 06-07)

responding L.app sets. Moreover, we can observe that the ratios of L. $\neg$ app are not so small; nearly half (46.1% and 40.4%) in the PTB III data, and more than half (70.7%, 54.3% and 64.3%) in CoNLL'00 and '03 data, respectively. This indicates that words not appearing in the labeled training data are really harmful for supervised learning. Although the performance with L. $\neg$ app sets is still poorer than with L.app sets, the JESS-CM results indicate that the introduction of unlabeled data effectively improves the performance of L. $\neg$ app sets, even more than that of L.app sets. These improvements are essentially very important; when a tagger and chunker are actually used, input data can be obtained from anywhere and this may mostly include words that do not appear in the given labeled training data since the labeled training data is limited and difficult to increase. This means that the improved performance of L. $\neg$ app can link directly to actual use.

Table 5 also shows the ratios of sentences that are constructed from words that all **appeared** in the 1G-word Unlabeled data used in our experiments (**U.app**) in the L. $\neg$ app and L.app. This indicates that most of the words in the development or test sets are covered by the 1G-word unlabeled data. This may be the main reason for JESS-CM providing large performance gains for both the overall and L. $\neg$ app set performance of all three tasks.

Table 6 shows the relation between JESS-CM performance and U.app in the NER experiments. The development data and test data were obtained from

system	dev.	test	additional resources
<b>JESS-CM</b> (CRF/HMM)	<b>97.35</b>	<b>97.40</b>	1G-word unlabeled data
(Shen et al., 2007)	97.28	97.33	–
(Toutanova et al., 2003)	97.15	97.24	crude company name detector
[sup. CRF (baseline)]	97.18	97.21	–

Table 7: POS tagging results of the previous top systems for PTB III data evaluated by label accuracy

system	test	additional resources
<b>JESS-CM</b> (CRF/HMM)	<b>95.15</b>	1G-word unlabeled data
	<b>94.67</b>	15M-word unlabeled data
(Ando and Zhang, 2005)	94.39	15M-word unlabeled data
(Suzuki et al., 2007)	94.36	17M-word unlabeled data
(Zhang et al., 2002)	94.17	full parser output
(Kudo and Matsumoto, 2001)	93.91	–
[supervised CRF (baseline)]	93.88	–

Table 8: Syntactic chunking results of the previous top systems for CoNLL'00 shared task data ( $F_{\beta=1}$  score)

30-31 Aug. 1996 and 6-7 Dec. 1996 Reuters news articles, respectively. We find that temporal proximity leads to better performance. This aspect can also be explained as U.app. Basically, the U.app increase leads to improved performance.

The evidence provided by the above experiments implies that increasing the coverage of unlabeled data offers the strong possibility of increasing the expected performance of unseen data. Thus, it strongly encourages us to use an SSL approach that includes JESS-CM to construct a general tagger and chunker for actual use.

## 5 Comparison with Previous Top Systems and Related Work

In POS tagging, the previous best performance was reported by (Shen et al., 2007) as summarized in Table 7. Their method uses a novel sophisticated model that learns both decoding order and labeling, while our model uses a standard first order Markov model. Despite using such a simple model, our method can provide a better result with the help of unlabeled data.

system	dev.	test	additional resources
<b>JESS-CM (CRF/HMM)</b>	<b>94.48</b>	<b>89.92</b>	1G-word unlabeled data
	93.66	<b>89.36</b>	37M-word unlabeled data
(Ando and Zhang, 2005)	93.15	89.31	27M-word unlabeled data
(Florian et al., 2003)	93.87	88.76	own large gazetteers, 2M-word labeled data
(Suzuki et al., 2007)	N/A	88.41	27M-word unlabeled data
[sup. CRF (baseline)]	91.74	86.35	-

Table 9: NER results of the previous top systems for CoNLL’03 shared task data evaluated by  $F_{\beta=1}$  score

As shown in Tables 8 and 9, the previous best performance for syntactic chunking and NER was reported by (Ando and Zhang, 2005), and is referred to as ‘ASO-semi’. ASO-semi also incorporates unlabeled data solely as additional information in the same way as JESS-CM. ASO-semi uses unlabeled data for constructing auxiliary problems that are expected to capture a good feature representation of the target problem. As regards syntactic chunking, JESS-CM significantly outperformed ASO-semi for the same 15M-word unlabeled data size obtained from the Wall Street Journal in 1991 as described in (Ando and Zhang, 2005). Unfortunately with NER, JESS-CM is slightly inferior to ASO-semi for the same 27M-word unlabeled data size extracted from the Reuters corpus. In fact, JESS-CM using 37M-words of unlabeled data provided a comparable result. We observed that ASO-semi prefers ‘nugget extraction’ tasks to ‘field segmentation’ tasks (Grenager et al., 2005). We cannot provide details here owing to the space limitation. Intuitively, their word prediction auxiliary problems can capture only a limited number of characteristic behaviors because the auxiliary problems are constructed by a limited number of ‘binary’ classifiers. Moreover, we should remember that ASO-semi used the human knowledge that ‘named entities mostly consist of nouns or adjectives’ during the auxiliary problem construction in their NER experiments. In contrast, our results require no such additional knowledge or limitation. In addition, the design and training of auxiliary problems as well as calculating SVD are too costly when the size of the unlabeled data increases. These facts imply that our SSL framework is rather appropriate for handling large scale unlabeled data.

On the other hand, ASO-semi and JESS-CM have an important common feature. That is, both meth-

ods discriminatively combine models trained by using unlabeled data in order to create informative feature representation for discriminative learning. Unlike self/co-training approaches (Blum and Mitchell, 1998), which use estimated labels as ‘correct labels’, this approach automatically judges the reliability of additional features obtained from unlabeled data in terms of discriminative training. Ando and Zhang (2007) have also pointed out that this methodology seems to be one key to achieving higher performance in NLP applications.

There is an approach that combines individually and independently trained joint PMs into a discriminative model (Li and McCallum, 2005). There is an essential difference between this method and JESS-CM. We categorize their approach as an ‘indirect approach’ since the outputs of the target task,  $\mathbf{y}$ , are not considered during the unlabeled data incorporation. Note that ASO-semi is also an ‘indirect approach’. On the other hand, our approach is a ‘direct approach’ because the distribution of  $\mathbf{y}$  obtained from JESS-CM is used as ‘seeds’ of hidden states during MDF estimation for joint PM parameters (see Section 4.1). In addition, MDF estimation over unlabeled data can effectively incorporate the ‘labeled’ training data information via a ‘bias’ since  $\lambda$  included in  $\mathcal{A}(\mathbf{x}, \mathbf{y})$  is estimated from labeled training data.

## 6 Conclusion

We proposed a simple yet powerful semi-supervised conditional model, which we call JESS-CM. It is applicable to large amounts of unlabeled data, for example, at the giga-word level. Experimental results obtained by using JESS-CM incorporating 1G-words of unlabeled data have provided the current best performance as regards POS tagging, syntactic chunking, and NER for widely used large test collections such as PTB III, CoNLL’00 and ’03 shared task data, respectively. We also provided evidence that the use of more unlabeled data in SSL can lead to further improvements. Moreover, our experimental analysis revealed that it may also induce an improvement in the expected performance for unseen data in terms of the unlabeled data coverage. Our results may encourage the adoption of the SSL method for many other real world applications.

## References

- R. Ando and T. Zhang. 2005. A High-Performance Semi-Supervised Learning Method for Text Chunking. In *Proc. of ACL-2005*, pages 1–9.
- R. Ando and T. Zhang. 2007. Two-view Feature Generation Model for Semi-supervised Learning. In *Proc. of ICML-2007*, pages 25–32.
- A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Conference on Computational Learning Theory 11*.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.
- R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. Named Entity Recognition through Classifier Combination. In *Proc. of CoNLL-2003*, pages 168–171.
- T. Grenager, D. Klein, and C. Manning. 2005. Unsupervised Learning of Field Segmentation Models for Information Extraction. In *Proc. of ACL-2005*, pages 371–378.
- T. Kudo and Y. Matsumoto. 2001. Chunking with Support Vector Machines. In *Proc. of NAACL 2001*, pages 192–199.
- J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of ICML-2001*, pages 282–289.
- W. Li and A. McCallum. 2005. Semi-Supervised Sequence Modeling with Syntactic Topic Models. In *Proc. of AAAI-2005*, pages 813–818.
- M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 2000. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39:103–134.
- F. Sha and F. Pereira. 2003. Shallow Parsing with Conditional Random Fields. In *Proc. of HLT/NAACL-2003*, pages 213–220.
- L. Shen, G. Satta, and A. Joshi. 2007. Guided Learning for Bidirectional Sequence Classification. In *Proc. of ACL-2007*, pages 760–767.
- C. Sutton, M. Sindelar, and A. McCallum. 2006. Reducing Weight Undertraining in Structured Discriminative Learning. In *Proc. of HLT-NAACL 2006*, pages 89–95.
- J. Suzuki, A. Fujino, and H. Isozaki. 2007. Semi-Supervised Structured Output Learning Based on a Hybrid Generative and Discriminative Approach. In *Proc. of EMNLP-CoNLL*, pages 791–800.
- E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proc. of CoNLL-2000 and LLL-2000*, pages 127–132.
- E. T. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proc. of CoNLL-2003*, pages 142–147.
- K. Toutanova, D. Klein, C.D. Manning, and Y. Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proc. of HLT-NAACL-2003*, pages 252–259.
- T. Zhang, F. Damerau, and D. Johnson. 2002. Text Chunking based on a Generalization of Winnow. *Machine Learning Research*, 2:615–637.

# Large Scale Acquisition of Paraphrases for Learning Surface Patterns

**Rahul Bhagat\***

Information Sciences Institute  
University of Southern California  
Marina del Rey, CA  
rahul@isi.edu

**Deepak Ravichandran**

Google Inc.  
1600 Amphitheatre Parkway  
Mountain View, CA  
deepakr@google.com

## Abstract

Paraphrases have proved to be useful in many applications, including Machine Translation, Question Answering, Summarization, and Information Retrieval. Paraphrase acquisition methods that use a single monolingual corpus often produce only syntactic paraphrases. We present a method for obtaining surface paraphrases, using a 150GB (25 billion words) monolingual corpus. Our method achieves an accuracy of around 70% on the paraphrase acquisition task. We further show that we can use these paraphrases to generate surface patterns for relation extraction. Our patterns are much more precise than those obtained by using a state of the art baseline and can extract relations with more than 80% precision for each of the test relations.

## 1 Introduction

Paraphrases are textual expressions that convey the same meaning using different surface words. For example consider the following sentences:

Google *acquired* YouTube. (1)

Google *completed the acquisition of* YouTube. (2)

Since they convey the same meaning, sentences (1) and (2) are sentence level paraphrases, and the phrases “*acquired*” and “*completed the acquisition of*” in (1) and (2) respectively are phrasal paraphrases.

Paraphrases provide a way to capture the variability of language and hence play an important

role in many natural language processing (NLP) applications. For example, in question answering, paraphrases have been used to find multiple patterns that pinpoint the same answer (Ravichandran and Hovy, 2002); in statistical machine translation, they have been used to find translations for unseen source language phrases (Callison-Burch et al., 2006); in multi-document summarization, they have been used to identify phrases from different sentences that express the same information (Barzilay et al., 1999); in information retrieval they have been used for query expansion (Anick and Tipirneni, 1999).

Learning paraphrases requires one to ensure identity of meaning. Since there are no adequate semantic interpretation systems available today, paraphrase acquisition techniques use some other mechanism as a kind of “pivot” to (help) ensure semantic identity. Each pivot mechanism selects phrases with similar meaning in a different characteristic way. A popular method, the so-called distributional similarity, is based on the dictum of Zelig Harris “you shall know the words by the company they keep”: given highly discriminating left and right contexts, only words with very similar meaning will be found to fit in between them. For paraphrasing, this has been often used to find syntactic transformations in parse trees that preserve (semantic) meaning. Another method is to use a bilingual dictionary or translation table as pivot mechanism: all source language words or phrases that translate to a given foreign word/phrase are deemed to be paraphrases of one another. In this paper we call the paraphrases that contain only words as surface paraphrases and those

---

\*Work done during an internship at Google Inc.

that contain paths in a syntax tree as syntactic paraphrases.

We here, present a method to acquire surface paraphrases from a single monolingual corpus. We use a large corpus (about 150GB) to overcome the data sparseness problem. To overcome the scalability problem, we pre-process the text with a simple parts-of-speech (POS) tagger and then apply locality sensitive hashing (LSH) (Charikar, 2002; Ravichandran et al., 2005) to speed up the remaining computation for paraphrase acquisition. Our experiments show results to verify the following main claim:

***Claim 1:** Highly precise surface paraphrases can be obtained from a very large monolingual corpus.*

With this result, we further show that these paraphrases can be used to obtain high precision surface patterns that enable the discovery of relations in a minimally supervised way. Surface patterns are templates for extracting information from text. For example, if one wanted to extract a list of company acquisitions, “⟨ACQUIRER⟩ *acquired* ⟨ACQUIREE⟩” would be one surface pattern with “⟨ACQUIRER⟩” and “⟨ACQUIREE⟩” as the slots to be extracted. Thus we can claim:

***Claim 2:** These paraphrases can then be used for generating high precision surface patterns for relation extraction.*

## 2 Related Work

Most recent work in paraphrase acquisition is based on automatic acquisition. Barzilay and McKeown (2001) used a monolingual parallel corpus to obtain paraphrases. Bannard and Callison-Burch (2005) and Zhou et al. (2006) both employed a bilingual parallel corpus in which each foreign language word or phrase was a pivot to obtain source language paraphrases. Dolan et al. (2004) and Barzilay and Lee (2003) used comparable news articles to obtain sentence level paraphrases. All these approaches rely on the presence of parallel or comparable corpora and are thus limited by their availability and size.

Lin and Pantel (2001) and Szpektor et al. (2004) proposed methods to obtain entailment templates by using a single monolingual resource. While both differ in their approaches, they both end up finding syntactic paraphrases. Their methods cannot be used if

we cannot parse the data (either because of scale or data quality). Our approach on the other hand, finds surface paraphrases; it is more scalable and robust due to the use of simple POS tagging. Also, our use of locality sensitive hashing makes finding similar phrases in a large corpus feasible.

Another task related to our work is relation extraction. Its aim is to extract instances of a given relation. Hearst (1992) the pioneering paper in the field used a small number of hand selected patterns to extract instances of hyponymy relation. Berland and Charniak (1999) used a similar method for extracting instances of meronymy relation. Ravichandran and Hovy (2002) used seed instances of a relation to automatically obtain surface patterns by querying the web. But their method often finds patterns that are too general (e.g., *X and Y*), resulting in low precision extractions. Rosenfeld and Feldman (2006) present a somewhat similar web based method that uses a combination of seed instances and seed patterns to learn good quality surface patterns. Both these methods differ from ours in that they learn relation patterns on the fly (from the web). Our method however, pre-computes paraphrases for a large set of surface patterns using distributional similarity over a large corpus and then obtains patterns for a relation by simply finding paraphrases (offline) for a few seed patterns. Using distributional similarity avoids the problem of obtaining overly general patterns and the pre-computation of paraphrases means that we can obtain the set of patterns for any relation instantaneously.

Romano et al. (2006) and Sekine (2006) used syntactic paraphrases to obtain patterns for extracting relations. While procedurally different, both methods depend heavily on the performance of the syntax parser and require complex syntax tree matching to extract the relation instances. Our method on the other hand acquires surface patterns and thus avoids the dependence on a parser and syntactic matching. This also makes the extraction process scalable.

## 3 Acquiring Paraphrases

This section describes our model for acquiring paraphrases from text.

### 3.1 Distributional Similarity

Harris’s distributional hypothesis (Harris, 1954) has played an important role in lexical semantics. It states that words that appear in similar contexts tend to have similar meanings. In this paper, we apply the distributional hypothesis to phrases i.e. word n-grams.

For example, consider the phrase “*acquired*” of the form “ $X$  *acquired*  $Y$ ”. Considering the context of this phrase, we might find {Google, eBay, Yahoo,...} in position  $X$  and {YouTube, Skype, Overture,...} in position  $Y$ . Now consider another phrase “*completed the acquisition of*”, again of the form “ $X$  *completed the acquisition of*  $Y$ ”. For this phrase, we might find {Google, eBay, Hilton Hotel corp.,...} in position  $X$  and {YouTube, Skype, Bally Entertainment Corp.,...} in position  $Y$ . Since the contexts of the two phrases are similar, our extension of the distributional hypothesis would assume that “*acquired*” and “*completed the acquisition of*” have similar meanings.

### 3.2 Paraphrase Learning Model

Let  $p$  be a phrase (n-gram) of the form  $X p Y$ , where  $X$  and  $Y$  are the placeholders for words occurring on either side of  $p$ . Our first task is to find the set of phrases that are similar in meaning to  $p$ . Let  $P = \{p_1, p_2, p_3, \dots, p_l\}$  be the set of all phrases of the form  $X p_i Y$  where  $p_i \in P$ . Let  $S_{i,X}$  be the set of words that occur in position  $X$  of  $p_i$  and  $S_{i,Y}$  be the set of words that occur in position  $Y$  of  $p_i$ . Let  $V_i$  be the vector representing  $p_i$  such that  $V_i = S_{i,X} \cup S_{i,Y}$ . Each word  $f \in V_i$  has an associated score that measures the strength of the association of the word  $f$  with phrase  $p_i$ ; as do many others, we employ pointwise mutual information (Cover and Thomas, 1991) to measure this strength of association.

$$pmi(p_i; f) = \log \frac{P(p_i, f)}{P(p_i)P(f)} \quad (1)$$

The probabilities in equation (1) are calculated by using the maximum likelihood estimate over our corpus.

Once we have the vectors for each phrase  $p_i \in P$ , we can find the paraphrases for each  $p_i$  by finding its nearest neighbors. We use cosine similarity, which

is a commonly used measure for finding similarity between two vectors.

If we have two phrases  $p_i \in P$  and  $p_j \in P$  with the corresponding vectors  $V_i$  and  $V_j$  constructed as described above, the similarity between the two phrases is calculated as:

$$sim(p_i; p_j) = \frac{V_i \cdot V_j}{|V_i| * |V_j|} \quad (2)$$

Each word in  $V_i$  (and  $V_j$ ) has with it an associated flag which indicates whether the word came from  $S_{i,X}$  or  $S_{i,Y}$ . Hence for each phrase  $p_i$  of the form  $X p_i Y$ , we have a corresponding phrase  $-p_i$  that has the form  $Y p_i X$ . This is important to find certain kinds of paraphrases. The following example will illustrate. Consider the sentences:

Google *acquired* YouTube. (3)

YouTube *was bought by* Google. (4)

From sentence (3), we obtain two phrases:

1.  $p_i =$  *acquired* which has the form “ $X$  *acquired*  $Y$ ” where “ $X =$  Google” and “ $Y =$  YouTube”
2.  $-p_i =$  *-acquired* which has the form “ $Y$  *acquired*  $X$ ” where “ $X =$  YouTube” and “ $Y =$  Google”

Similarly, from sentence (4) we obtain two phrases:

1.  $p_j =$  *was bought by* which has the form “ $X$  *was bought by*  $Y$ ” where “ $X =$  YouTube” and “ $Y =$  Google”
2.  $-p_j =$  *-was bought by* which has the form “ $Y$  *was bought by*  $X$ ” where “ $X =$  Google” and “ $Y =$  YouTube”

The switching of  $X$  and  $Y$  positions in (3) and (4) ensures that “*acquired*” and “*-was bought by*” are found to be paraphrases by the algorithm.

### 3.3 Locality Sensitive Hashing

As described in Section 3.2, we find paraphrases of a phrase  $p_i$  by finding its nearest neighbors based on cosine similarity between the feature vector of  $p_i$  and other phrases. To do this for all the phrases in the corpus, we’ll have to compute the similarity between all vector pairs. If  $n$  is the number of vectors and  $d$  is the dimensionality of the vector space, finding cosine similarity between each pair of vectors has time complexity  $O(n^2 d)$ . This computation is infeasible for our corpus, since both  $n$  and  $d$  are large.



To solve this problem, we make use of Locality Sensitive Hashing (LSH). The basic idea behind LSH is that a LSH function creates a fingerprint for each vector such that if two vectors are similar, they are likely to have similar fingerprints. The LSH function we use here was proposed by Charikar (2002). It represents a  $d$  dimensional vector by a stream of  $b$  bits ( $b \ll d$ ) and has the property of preserving the cosine similarity between vectors, which is exactly what we want. Ravichandran et al. (2005) have shown that by using the LSH nearest neighbors calculation can be done in  $O(nd)$  time.<sup>1</sup>

## 4 Learning Surface Patterns

Let  $r$  be a target relation. Our task is to find a set of surface patterns  $S = \{s_1, s_2, \dots, s_n\}$  that express the target relation. For example, consider the relation  $r = \text{“acquisition”}$ . We want to find the set of patterns  $S$  that express this relation:

$$S = \{ \langle \text{ACQUIRER} \rangle \text{ acquired } \langle \text{ACQUIREE} \rangle, \langle \text{ACQUIRER} \rangle \text{ bought } \langle \text{ACQUIREE} \rangle, \langle \text{ACQUIREE} \rangle \text{ was bought by } \langle \text{ACQUIRER} \rangle, \dots \}.$$

The remainder of the section describes our model for learning surface patterns for target relations.

### 4.1 Model Assumption

Paraphrases express the same meaning using different surface forms. So if one knew a pattern that expresses a target relation, one could build more patterns for that relation by finding paraphrases for the surface phrase(s) in that pattern. This is the basic assumption of our model.

For example, consider the seed pattern “ $\langle \text{ACQUIRER} \rangle$  *acquired*  $\langle \text{ACQUIREE} \rangle$ ” for the target relation “*acquisition*”. The surface phrase in the seed pattern is “*acquired*”. Our model then assumes that we can obtain more surface patterns for “*acquisition*” by replacing “*acquired*” in the seed pattern with its paraphrases i.e.  $\{ \text{bought}, -\text{was bought by}^2, \dots \}$ . The resulting surface patterns are:

<sup>1</sup>The details of the algorithm are omitted, but interested readers are encouraged to read Charikar (2002) and Ravichandran et al. (2005)

<sup>2</sup>The “-” in “-was bought by” indicates that the  $\langle \text{ACQUIRER} \rangle$  and  $\langle \text{ACQUIREE} \rangle$  arguments of the input phrase “*acquired*” need to be switched for the phrase “was bought by”.

$$\{ \langle \text{ACQUIRER} \rangle \text{ bought } \langle \text{ACQUIREE} \rangle, \langle \text{ACQUIREE} \rangle \text{ was bought by } \langle \text{ACQUIRER} \rangle, \dots \}$$

### 4.2 Surface Pattern Model

Let  $r$  be a target relation. Let  $SEED = \{seed_1, seed_2, \dots, seed_n\}$  be the set of seed patterns that express the target relation. For each  $seed_i \in SEED$ , we obtain the corresponding set of new patterns  $PAT_i$  in two steps:

1. We find the surface phrase,  $p_i$ , using a seed and find the corresponding set of paraphrases,  $P_i = \{p_{i,1}, p_{i,2}, \dots, p_{i,m}\}$ . Each paraphrase,  $p_{i,j} \in P_i$ , has with it an associated score which is similarity between  $p_i$  and  $p_{i,j}$ .
2. In seed pattern,  $seed_i$ , we replace the surface phrase,  $p_i$ , with its paraphrases and obtain the set of new patterns  $PAT_i = \{pat_{i,1}, pat_{i,2}, \dots, pat_{i,m}\}$ . Each pattern has with it an associated score, which is the same as the score of the paraphrase from which it was obtained<sup>3</sup>. The patterns are ranked in the decreasing order of their scores.

After we obtain  $PAT_i$  for each  $seed_i \in SEED$ , we obtain the complete set of patterns,  $PAT$ , for the target relation  $r$  as the union of all the individual pattern sets, i.e.,  $PAT = PAT_1 \cup PAT_2 \cup \dots \cup PAT_n$ .

## 5 Experimental Methodology

In this section, we describe experiments to validate the main claims of the paper. We first describe paraphrase acquisition, we then summarize our method for learning surface patterns, and finally describe the use of patterns for extracting relation instances.

### 5.1 Paraphrases

Finding surface variations in text requires a large corpus. The corpus needs to be orders of magnitude larger than that required for learning syntactic variations, since surface phrases are sparser than syntactic phrases.

For our experiments, we used a corpus of about 150GB (25 billion words) obtained from Google News<sup>4</sup>. It consists of few years worth of news data.

<sup>3</sup>If a pattern is generated from more than one seed, we assign it its average score.

<sup>4</sup>The corpus was cleaned to remove duplicate articles.

We POS tagged the corpus using Tnt tagger (Brants, 2000) and collected all phrases (n-grams) in the corpus that contained at least one verb, and had a noun or a noun-noun compound on either side. We restricted the phrase length to at most five words.

We build a vector for each phrase as described in Section 3. To mitigate the problem of sparseness and co-reference to a certain extent, whenever we have a noun-noun compound in the  $X$  or  $Y$  positions, we treat it as bag of words. For example, in the sentence “Google Inc. *acquired* YouTube”, “Google” and “Inc.” will be treated as separate features in the vector<sup>5</sup>.

Once we have constructed all the vectors, we find the paraphrases for every phrase by finding its nearest neighbors as described in Section 3. For our experiments, we set the number of random bits in the LSH function to 3000, and the similarity cut-off between vectors to 0.15. We eventually end up with a resource containing over 2.5 million phrases such that each phrase is connected to its paraphrases.

## 5.2 Surface Patterns

One claim of this paper is that we can find good surface patterns for a target relation by starting with a seed pattern. To verify this, we study two target relations<sup>6</sup>:

1. **Acquisition:** *We define this as the relation between two companies such that one company acquired the other.*
2. **Birthplace:** *We define this as the relation between a person and his/her birthplace.*

For “*acquisition*” relation, we start with the surface patterns containing only the words *buy* and *acquire*:

1. “⟨ACQUIRER⟩ *bought* ⟨ACQUIREE⟩” (and its variants, i.e. *buy*, *buys* and *buying*)
2. “⟨ACQUIRER⟩ *acquired* ⟨ACQUIREE⟩” (and its variants, i.e. *acquire*, *acquires* and *acquiring*)

<sup>5</sup>This adds some noise in the vectors, but we found that this results in better paraphrases.

<sup>6</sup>Since we have to do all the annotations for evaluations on our own, we restricted our experiments to only two commonly used relations.

This results in a total of eight seed patterns.

For “*birthplace*” relation, we start with two seed patterns:

1. “⟨PERSON⟩ *was born in* ⟨LOCATION⟩”
2. “⟨PERSON⟩ *was born at* ⟨LOCATION⟩”.

We find other surface patterns for each of these relations by replacing the surface words in the seed patterns by their paraphrases, as described in Section 4.

## 5.3 Relation Extraction

The purpose of learning surface patterns for a relation is to extract instances of that relation. We use the surface patterns obtained for the relations “*acquisition*” and “*birthplace*” to extract instances of these relations from the LDC North American News Corpus. This helps us to extrinsically evaluate the quality of the surface patterns.

## 6 Experimental Results

In this section, we present the results of the experiments and analyze them.

### 6.1 Baselines

It is hard to construct a baseline for comparing the quality of paraphrases, as there isn’t much work in extracting surface level paraphrases using a monolingual corpus. To overcome this, we show the effect of reduction in corpus size on the quality of paraphrases, and compare the results informally to the other methods that produce syntactic paraphrases.

To compare the quality of the extraction patterns, and relation instances, we use the method presented by Ravichandran and Hovy (2002) as the baseline. For each of the given relations, “*acquisition*” and “*birthplace*”, we use 10 seed instances, download the top 1000 results from the Google search engine for each instance, extract the sentences that contain the instances, and learn the set of baseline patterns for each relation. We then apply these patterns to the test corpus and extract the corresponding baseline instances.

### 6.2 Evaluation Criteria

Here we present the evaluation criteria we used to evaluate the performance on the different tasks.

## Paraphrases

We estimate the quality of paraphrases by annotating a random sample as correct/incorrect and calculating the accuracy. However, estimating the recall is difficult given that we do not have a complete set of paraphrases for the input phrases. Following Szpektor et al. (2004), instead of measuring recall, we calculate the average number of correct paraphrases per input phrase.

## Surface Patterns

We can calculate the precision ( $P$ ) of learned patterns for each relation by annotating the extracted patterns as correct/incorrect. However calculating the recall is a problem for the same reason as above. But we can calculate the relative recall ( $RR$ ) of the system against the baseline and vice versa. The relative recall  $RR_{S|B}$  of system  $S$  with respect to system  $B$  can be calculated as:

$$RR_{S|B} = \frac{C_S \cap C_B}{C_B}$$

where  $C_S$  is the number of correct patterns found by our system and  $C_B$  is the number of correct patterns found by the baseline.  $RR_{B|S}$  can be found in a similar way.

## Relation Extraction

We estimate the precision ( $P$ ) of the extracted instances by annotating a random sample of instances as correct/incorrect. While calculating the true recall here is not possible, even calculating the true relative recall of the system against the baseline is not possible as we can annotate only a small sample. However, following Pantel et al. (2004), we assume that the recall of the baseline is 1 and estimate the relative recall  $RR_{S|B}$  of the system  $S$  with respect to the baseline  $B$  using their respective precision scores  $P_S$  and  $P_B$  and number of instances extracted by them  $|S|$  and  $|B|$  as:

$$RR_{S|B} = \frac{P_S * |S|}{P_B * |B|}$$

## 6.3 Gold Standard

In this section, we describe the creation of gold standard for the different tasks.

### Paraphrases

We created the gold standard paraphrase test set by randomly selecting 50 phrases and their corresponding paraphrases from our collection of 2.5 million

phrases. For each test phrase, we asked two annotators to annotate its paraphrases as correct/incorrect. The annotators were instructed to look for strict paraphrases i.e. equivalent phrases that can be substituted for each other.

To obtain the inter-annotator agreement, the two annotators annotated the test set separately. The kappa statistic (Siegal and Castellan Jr., 1988) was  $\kappa = 0.63$ . The interesting thing is that the annotators got this respectable kappa score without any prior training, which is hard to achieve when one annotates for a similar task like textual entailment.

### Surface Patterns

For the target relations, we asked two annotators to annotate the patterns for each relation as either “precise” or “vague”. The annotators annotated the system as well as the baseline outputs. We consider the “precise” patterns as correct and the “vague” as incorrect. The intuition is that applying the vague patterns for extracting target relation instances might find some good instances, but will also find many bad ones. For example, consider the following two patterns for the “*acquisition*” relation:

⟨ACQUIRER⟩ *acquired* ⟨ACQUIREE⟩ (5)

⟨ACQUIRER⟩ *and* ⟨ACQUIREE⟩ (6)

Example (5) is a precise pattern as it clearly identifies the “*acquisition*” relation while example (6) is a vague pattern because it is too general and says nothing about the “*acquisition*” relation. The kappa statistic between the two annotators for this task was  $\kappa = 0.72$ .

### Relation Extraction

We randomly sampled 50 instances of the “*acquisition*” and “*birthplace*” relations from the system and the baseline outputs. We asked two annotators to annotate the instances as correct/incorrect. The annotators marked an instance as correct only if both the entities and the relation between them were correct. To make their task easier, the annotators were provided the context for each instance, and were free to use any resources at their disposal (including a web search engine), to verify the correctness of the instances. The annotators found that the annotation for this task was much easier than the previous two; the few disagreements they had were due to ambiguity of some of the instances. The kappa statistic for this task was  $\kappa = 0.91$ .

Annotator	Accuracy	Average # correct paraphrases
Annotator 1	67.31%	4.2
Annotator 2	74.27%	4.28

Table 1: Quality of paraphrases

are being distributed to	approved a revision to the
have been distributed to	unanimously approved a new
are being handed out to	approved an annual
were distributed to	will consider adopting a
–are handing out	approved a revised
will be distributed to all	approved a new

Table 2: Example paraphrases

## 6.4 Result Summary

Table 1 shows the results of annotating the paraphrases test set. We do not have a baseline to compare against but we can analyze them in light of numbers reported previously for syntactic paraphrases. DIRT (Lin and Pantel, 2001) and TEASE (Szpektor et al., 2004) report accuracies of 50.1% and 44.3% respectively compared to our average accuracy across two annotators of 70.79%. The average number of paraphrases per phrase is however 10.1 and 5.5 for DIRT and TEASE respectively compared to our 4.2. One reason why this number is lower is that our test set contains completely random phrases from our set (2.5 million phrases): some of these phrases are rare and have very few paraphrases. Table 2 shows some paraphrases generated by our system for the phrases “*are being distributed to*” and “*approved a revision to the*”.

Table 3 shows the results on the quality of surface patterns for the two relations. It can be observed that our method outperforms the baseline by a wide margin in both precision and relative recall. Table 4 shows some example patterns learned by our system.

Table 5 shows the results of the quality of extracted instances. Our system obtains very high precision scores but suffers in relative recall given that the baseline with its very general patterns is likely to find a huge number of instances (though a very small portion of them are correct). Table 6 shows some example instances we extracted.

acquisition	birthplace
<i>X agreed to buy Y</i>	<i>X , who was born in Y</i>
<i>X , which acquired Y</i>	<i>X , was born in Y</i>
<i>X completed its acquisition of Y</i>	<i>X was raised in Y</i>
<i>X has acquired Y</i>	<i>X was born in NNNN<sup>a</sup> in Y</i>
<i>X purchased Y</i>	<i>X , born in Y</i>

<sup>a</sup>Each “N” here is a placeholder for a number from 0 to 9.

Table 4: Example extraction templates

acquisition	birthplace
1. <i>Huntington Bancshares Inc. agreed to acquire Reliance Bank</i>	1. <i>Cyril Andrew Ponnampereuma was born in Galle</i>
2. <i>Sony bought Columbia Pictures</i>	2. <i>Cook was born in NNNN in Devonshire</i>
3. <i>Hanson Industries buys Kidde Inc.</i>	3. <i>Tansey was born in Cincinnati</i>
4. <i>Casino America inc. agreed to buy Grand Palais</i>	4. <i>Tsoi was born in NNNN in Uzbekistan</i>
5. <i>Tidewater inc. acquired Hornbeck Offshore Services Inc.</i>	5. <i>Mrs. Totenberg was born in San Francisco</i>

Table 6: Example instances

## 6.5 Discussion and Error Analysis

We studied the effect of the decrease in size of the available raw corpus on the quality of the acquired paraphrases. We used about 10% of our original corpus to learn the surface paraphrases and evaluated them. The precision, and the average number of correct paraphrases are calculated on the same test set, as described in Section 6.2. The performance drop on using 10% of the original corpus is significant (11.41% precision and on an average 1 correct paraphrase per phrase), which shows that we indeed need a large amount of data to learn good quality surface paraphrases. One reason for this drop is also that when we use only 10% of the original data, for some of the phrases from the test set, we do not find any paraphrases (thus resulting in 0% accuracy for them). This is not unexpected, as the larger resource would have a much larger recall, which again points at the advantage of using a large data set. Another reason for this performance drop could be the parameter settings: We found that the quality of learned paraphrases depended greatly on the various cut-offs used. While we adjusted our model

Relation	Method	# Patterns	Annotator 1		Annotator 2	
			<i>P</i>	<i>RR</i>	<i>P</i>	<i>RR</i>
Acquisition	Baseline	160	55%	13.02%	60%	11.16%
	Paraphrase Method	231	<b>83.11%</b>	<b>28.40%</b>	<b>93.07%</b>	<b>25%</b>
Birthplace	Baseline	16	31.35%	15.38%	31.25%	15.38%
	Paraphrase Method	16	<b>81.25%</b>	<b>40%</b>	<b>81.25%</b>	<b>40%</b>

Table 3: Quality of Extraction Patterns

Relation	Method	# Patterns	Annotator 1		Annotator 2	
			<i>P</i>	<i>RR</i>	<i>P</i>	<i>RR</i>
Acquisition	Baseline	1,261,986	6%	<b>100%</b>	2%	<b>100%</b>
	Paraphrase Method	3875	<b>88%</b>	4.5%	<b>82%</b>	12.59%
Birthplace	Baseline	979,607	4%	<b>100%</b>	2%	<b>100%</b>
	Paraphrase Method	1811	<b>98%</b>	4.53%	<b>98%</b>	9.06%

Table 5: Quality of instances

parameters for working with smaller sized data, it is conceivable that we did not find the ideal setting for them. So we consider these numbers to be a lower bound. But even then, these numbers clearly indicate the advantage of using more data.

We also manually inspected our paraphrases. We found that the problem of “antonyms” was somewhat less pronounced due to our use of a large corpus, but they still were the major source of error. For example, our system finds the phrase “*sell*” as a paraphrase for “*buy*”. We need to deal with this problem separately in the future (may be as a post-processing step using a list of antonyms).

Moving to the task of relation extraction, we see from table 5 that our system has a much lower relative recall compared to the baseline. This was expected as the baseline method learns some very general patterns, which are likely to extract some good instances, even though they result in a huge hit to its precision. However, our system was able to obtain this performance using very few seeds. So an increase in the number of input seeds, is likely to increase the relative recall of the resource. The question however remains as to what good seeds might be. It is clear that it is much harder to come up with good seed patterns (that our system needs), than seed instances (that the baseline needs). But there are some obvious ways to overcome this problem. One way is to bootstrap. We can look at the paraphrases of the seed patterns and use them to obtain more patterns. Our initial experiments with this method using handpicked seeds showed good promise. However,

we need to investigate automating this approach. Another method is to use the good patterns from the baseline system and use them as seeds for our system. We plan to investigate this approach as well. One reason, why we have seen good preliminary results using these approaches (for improving recall), we believe, is that the precision of the paraphrases is good. So either a seed doesn’t produce any new patterns or it produces good patterns, thus keeping the precision of the system high while increasing relative recall.

## 7 Conclusion

Paraphrases are an important technique to handle variations in language. Given their utility in many NLP tasks, it is desirable that we come up with methods that produce good quality paraphrases. We believe that the paraphrase acquisition method presented here is a step towards this very goal. We have shown that high precision surface paraphrases can be obtained by using distributional similarity on a large corpus. We made use of some recent advances in theoretical computer science to make this task scalable. We have also shown that these paraphrases can be used to obtain high precision extraction patterns for information extraction. While we believe that more work needs to be done to improve the system recall (some of which we are investigating), this seems to be a good first step towards developing a minimally supervised, easy to implement, and scalable relation extraction system.

## References

- P. G. Anick and S. Tipirneni. 1999. The paraphrase search assistant: terminological feedback for iterative information seeking. In *ACM SIGIR*, pages 153–159.
- C. Bannard and C. Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Association for Computational Linguistics*, pages 597–604.
- R. Barzilay and L. Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *In Proceedings North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 16–23.
- R. Barzilay and K. R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *In Proceedings of Association for Computational Linguistics*, pages 50–57.
- R. Barzilay, K. R. McKeown, and M. Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Association for Computational Linguistics*, pages 550–557.
- M. Berland and E. Charniak. 1999. Finding parts in very large corpora. In *In Proceedings of Association for Computational Linguistics*, pages 57–64.
- T. Brants. 2000. Tnt – a statistical part-of-speech tagger. In *In Proceedings of the Applied NLP Conference (ANLP)*.
- C. Callison-Burch, P. Koehn, and M. Osborne. 2006. Improved statistical machine translation using paraphrases. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 17–24.
- M. S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *In Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 380–388.
- T.M. Cover and J.A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons.
- B. Dolan, C. Quirk, and C. Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *In Proceedings of the conference on Computational Linguistics (COLING)*, pages 350–357.
- Z. Harris. 1954. Distributional structure. *Word*, pages 10(23):146–162.
- M. A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the conference on Computational linguistics*, pages 539–545.
- D. Lin and P. Pantel. 2001. Dirt: Discovery of inference rules from text. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328.
- P. Pantel, D. Ravichandran, and E.H. Hovy. 2004. Towards terascale knowledge acquisition. In *In Proceedings of the conference on Computational Linguistics (COLING)*, pages 771–778.
- D. Ravichandran and E.H. Hovy. 2002. Learning surface text for a question answering system. In *Association for Computational Linguistics (ACL)*, Philadelphia, PA.
- D. Ravichandran, P. Pantel, and E.H. Hovy. 2005. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *In Proceedings of Association for Computational Linguistics*, pages 622–629.
- L. Romano, M. Kouylekov, I. Szpektor, I. Dagan, and A. Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *In Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.
- B. Rosenfeld and R. Feldman. 2006. Ures: an unsupervised web relation extraction system. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 667–674.
- S. Sekine. 2006. On-demand information extraction. In *In Proceedings of COLING/ACL*, pages 731–738.
- S. Siegal and N.J. Castellan Jr. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill.
- I. Szpektor, H. Tanev, I. Dagan, and B. Coppola. 2004. Scaling web-based acquisition of entailment relations. In *In Proceedings of Empirical Methods in Natural Language Processing*, pages 41–48.
- L. Zhou, C.Y. Lin, D. Munteanu, and E.H. Hovy. 2006. Paraeval: using paraphrases to evaluate summaries automatically. In *In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 447–454.

# Contextual Preferences

**Idan Szpektor, Ido Dagan, Roy Bar-Haim**

Department of Computer Science

Bar-Ilan University

Ramat Gan, Israel

{szpekti,dagan,barhair}@cs.biu.ac.il

**Jacob Goldberger**

School of Engineering

Bar-Ilan University

Ramat Gan, Israel

goldbej@eng.biu.ac.il

## Abstract

The validity of semantic inferences depends on the contexts in which they are applied. We propose a generic framework for handling contextual considerations within applied inference, termed *Contextual Preferences*. This framework defines the various context-aware components needed for inference and their relationships. Contextual preferences extend and generalize previous notions, such as selectional preferences, while experiments show that the extended framework allows improving inference quality on real application data.

## 1 Introduction

Applied semantic inference is typically concerned with inferring a target meaning from a given text. For example, to answer “*Who wrote Idomeneo?*”, Question Answering (QA) systems need to infer the target meaning ‘Mozart wrote Idomeneo’ from a given text “*Mozart composed Idomeneo*”. Following common Textual Entailment terminology (Giampiccolo et al., 2007), we denote the target meaning by  $h$  (for *hypothesis*) and the given text by  $t$ .

A typical applied inference operation is *matching*. Sometimes,  $h$  can be directly matched in  $t$  (in the example above, if the given sentence would be literally “*Mozart wrote Idomeneo*”). Generally, the target meaning can be expressed in  $t$  in many different ways. Indirect matching is then needed, using inference knowledge that may be captured through rules, termed here *entailment rules*. In our example, ‘Mozart wrote Idomeneo’ can be inferred using the rule ‘ $X$  compose  $Y \rightarrow X$  write  $Y$ ’. Recently,

several algorithms were proposed for automatically learning entailment rules and paraphrases (viewed as bi-directional entailment rules) (Lin and Pantel, 2001; Ravichandran and Hovy, 2002; Shinyama et al., 2002; Szpektor et al., 2004; Sekine, 2005).

A common practice is to try matching the structure of  $h$ , or of the left-hand-side of a rule  $r$ , within  $t$ . However, context should be considered to allow valid matching. For example, suppose that to find acquisitions of companies we specify the target *template hypothesis* (a hypothesis with variables) ‘ $X$  acquire  $Y$ ’. This  $h$  should not be matched in “*children acquire language quickly*”, because in this context  $Y$  is not a company. Similarly, the rule ‘ $X$  charge  $Y \rightarrow X$  accuse  $Y$ ’ should not be applied to “*This store charged my account*”, since the assumed sense of ‘charge’ in the rule is different than its sense in the text. Thus, the intended contexts for  $h$  and  $r$  and the context within the given  $t$  should be properly matched to verify valid inference.

Context matching at inference time was often approached in an application-specific manner (Harabagiu et al., 2003; Patwardhan and Riloff, 2007). Recently, some generic methods were proposed to handle context-sensitive inference (Dagan et al., 2006; Pantel et al., 2007; Downey et al., 2007; Connor and Roth, 2007), but these usually treat only a single aspect of context matching (see Section 6). We propose a comprehensive framework for handling various contextual considerations, termed *Contextual Preferences*. It extends and generalizes previous work, defining the needed contextual components and their relationships. We also present and implement concrete representation models and un-

supervised matching methods for these components. While our presentation focuses on semantic inference using lexical-syntactic structures, the proposed framework and models seem suitable for other common types of representations as well.

We applied our models to a test set derived from the ACE 2005 event detection task, a standard Information Extraction (IE) benchmark. We show the benefits of our extended framework for textual inference and present component-wise analysis of the results. To the best of our knowledge, these are also the first unsupervised results for event argument extraction in the ACE 2005 dataset.

## 2 Contextual Preferences

### 2.1 Notation

As mentioned above, we follow the generic Textual Entailment (TE) setting, testing whether a target meaning hypothesis  $h$  can be inferred from a given text  $t$ . We allow  $h$  to be either a text or a *template*, a text fragment with variables. For example, “*The stock rose 8%*” entails an instantiation of the template hypothesis ‘ $X$  gain  $Y$ ’. Typically,  $h$  represents an information need requested in some application, such as a target predicate in IE.

In this paper, we focus on parse-based lexical-syntactic representation of texts and hypotheses, and on the basic inference operation of *matching*. Following common practice (de Salvo Braz et al., 2005; Romano et al., 2006; Bar-Haim et al., 2007),  $h$  is syntactically matched in  $t$  if it can be embedded in  $t$ ’s parse tree. For template hypotheses, the matching induces a mapping between  $h$ ’s variables and their instantiation in  $t$ .

Matching  $h$  in  $t$  can be performed either directly or indirectly using entailment rules. An *entailment rule*  $r$ : ‘ $LHS \rightarrow RHS$ ’ is a directional entailment relation between two templates.  $h$  is matched in  $t$  using  $r$  if  $LHS$  is matched in  $t$  and  $h$  matches  $RHS$ . In the example above,  $r$ : ‘ $X$  rise  $Y \rightarrow X$  gain  $Y$ ’ allows us to entail ‘ $X$  gain  $Y$ ’, with “stock” and “8%” instantiating  $h$ ’s variables. We denote  $vars(z)$  the set of variables of  $z$ , where  $z$  is a template or a rule.

### 2.2 Motivation

When matching considers only the structure of hypotheses, texts and rules it may result in incorrect

inference due to contextual mismatches. For example, an IE system may identify mentions of public demonstrations using the hypothesis  $h$ : ‘ $X$  demonstrate’ . However,  $h$  should not be matched in “*Engineers demonstrated the new system*”, due to a mismatch between the intended sense of ‘demonstrate’ in  $h$  and its sense in  $t$ . Similarly, when looking for physical attack mentions using the hypothesis ‘ $X$  attack  $Y$ ’, we should not utilize the rule  $r$ : ‘ $X$  accuse  $Y \rightarrow X$  attack  $Y$ ’, due to a mismatch between a verbal attack in  $r$  and an intended physical attack in  $h$ . Finally,  $r$ : ‘ $X$  produce  $Y \rightarrow X$  lay  $Y$ ’ (applicable when  $X$  refers to poultry and  $Y$  to eggs) should not be matched in  $t$ : “*Bugatti produce the fastest cars*”, due to a mismatch between the meanings of ‘produce’ in  $r$  and  $t$ . Overall, such incorrect inferences may be avoided by considering contextual information for  $t$ ,  $h$  and  $r$  during their matching process.

### 2.3 The Contextual Preferences Framework

We propose the *Contextual Preferences* (CP) framework for addressing context at inference time. In this framework, the representation of an object  $z$ , where  $z$  may be a text, a template or an entailment rule, is enriched with contextual information denoted  $cp(z)$ . This information helps constraining or disambiguating the meaning of  $z$ , and is used to validate proper matching between pairs of objects.

We consider two components within  $cp(z)$ : (a) a representation for the global (“topical”) context in which  $z$  typically occurs, denoted  $cp_g(z)$ ; (b) a representation for the preferences and constraints (“hard” preferences) on the possible terms that can instantiate variables within  $z$ , denoted  $cp_v(z)$ . For example,  $cp_v(‘X$  produce  $Y \rightarrow X$  lay  $Y$ ’) may specify that  $X$ ’s instantiations should be similar to “chicken” or “duck”.

Contextual Preferences are used when entailment is assessed between a text  $t$  and a hypothesis  $h$ , either directly or by utilizing an entailment-rule  $r$ . On top of structural matching, we now require that the Contextual Preferences of the participants in the inference will also match. When  $h$  is directly matched in  $t$ , we require that each component in  $cp(h)$  will be matched with its counterpart in  $cp(t)$ . When  $r$  is utilized, we additionally require that  $cp(r)$  will be matched with both  $cp(t)$  and  $cp(h)$ . Figure 1 summarizes the matching relationships between the CP



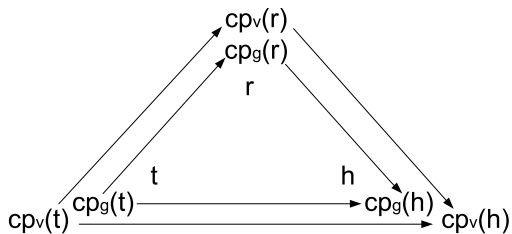


Figure 1: The directional matching relationships between a hypothesis ( $h$ ), an entailment rule ( $r$ ) and a text ( $t$ ) in the Contextual Preferences framework.

components of  $h$ ,  $t$  and  $r$ .

Like Textual Entailment inference, Contextual Preferences matching is directional. When matching  $h$  with  $t$  we require that the global context preferences specified by  $cp_g(h)$  would subsume those induced by  $cp_g(t)$ , and that the instantiations of  $h$ 's variables in  $t$  would adhere to the preferences in  $cp_v(h)$  (since  $t$  should entail  $h$ , but not necessarily vice versa). For example, if the preferred global context of a hypothesis is sports, it would match a text that discusses the more specific topic of basketball.

To implement the CP framework, concrete models are needed for each component, specifying its representation, how it is constructed, and an appropriate matching procedure. Section 3 describes the specific CP models that were implemented in this paper.

The CP framework provides a generic view of contextual modeling in applied semantic inference. Mapping from a specific application to the generic framework follows the mappings assumed in the Textual Entailment paradigm. For example, in QA the hypothesis to be proved corresponds to the affirmative template derived from the question (e.g.  $h$ : 'X invented the PC' for "Who invented the PC?"). Thus,  $cp_g(h)$  can be constructed with respect to the question's focus while  $cp_v(h)$  may be generated from the expected answer type (Moldovan et al., 2000; Harabagiu et al., 2003). Construction of hypotheses' CP for IE is demonstrated in Section 4.

### 3 Contextual Preferences Models

This section presents the current models that we implemented for the various components of the CP framework. For each component type we describe its representation, how it is constructed, and a cor-

responding unsupervised match score. Finally, the different component scores are combined to yield an overall match score, which is used in our experiments to rank inference instances by the likelihood of their validity. Our goal in this paper is to cover the entire scope of the CP framework by including specific models that were proposed in previous work, where available, and elsewhere propose initial models to complete the CP scope.

#### 3.1 Contextual Preferences for Global Context

To represent the global context of an object  $z$  we utilize Latent Semantic Analysis (LSA) (Deerwester et al., 1990), a well-known method for representing the contextual-usage of words based on corpus statistics. We use LSA analysis of the BNC corpus<sup>1</sup>, in which every term is represented by a normalized vector of the top 100 SVD dimensions, as described in (Gliozzo, 2005).

To construct  $cp_g(z)$  we first collect a set of terms that are representative for the preferred general context of  $z$ . Then, the (single) vector which is the sum of the LSA vectors of the representative terms becomes the representation of  $cp_g(z)$ . This LSA vector captures the "average" typical contexts in which the representative terms occur.

The set of representative terms for a text  $t$  consists of all the nouns and verbs in it, represented by their lemma and part of speech. For a rule  $r$ : ' $LHS \rightarrow RHS$ ', the representative terms are the words appearing in  $LHS$  and in  $RHS$ . For example, the representative terms for ' $X$  divorce  $Y \rightarrow X$  marry  $Y$ ' are  $\{divorce:v, marry:v\}$ . As mentioned earlier, construction of hypotheses and their contextual preferences depends on the application at hand. In our experiments these are defined manually, as described in Section 4, derived from the manual definitions of target meanings in the IE data.

The score of matching the  $cp_g$  components of two objects, denoted by  $m_g(\cdot, \cdot)$ , is the Cosine similarity of their LSA vectors. Negative values are set to 0.

#### 3.2 Contextual Preferences for Variables

##### 3.2.1 Representation

For comparison with prior work, we follow (Pantel et al., 2007) and represent preferences for vari-

<sup>1</sup><http://www.natcorp.ox.ac.uk/>

able instantiations using a distributional approach, and in addition incorporate a standard specification of named-entity types. Thus,  $cp_v$  is represented by two lists. The first list, denoted  $cp_{v:e}$ , contains examples for valid instantiations of that variable. For example,  $cp_{v:e}(X \text{ kill } Y \rightarrow Y \text{ die of } X)$  may be  $[X: \{\textit{snakebite}, \textit{disease}\}, Y: \{\textit{man}, \textit{patient}\}]$ . The second list, denoted  $cp_{v:n}$ , contains the variable’s preferred named-entity types (if any). For example,  $cp_{v:n}(X \text{ born in } Y)$  may be  $[X: \{\textit{Person}\}, Y: \{\textit{Location}\}]$ . We denote  $cp_{v:e}(z)[j]$  and  $cp_{v:n}(z)[j]$  as the lists for a specific variable  $j$  of the object  $z$ .

For a text  $t$ , in which a template  $p$  is matched, the preference  $cp_{v:e}(t)$  for each template variable is simply its instantiation in  $t$ . For example, when ‘ $X$  eat  $Y$ ’ is matched in  $t$ : “*Many Americans eat fish regularly*”, we construct  $cp_{v:e}(t) = [X: \{\textit{Many Americans}\}, Y: \{\textit{fish}\}]$ . Similarly,  $cp_{v:n}(t)$  for each variable is the named-entity type of its instantiation in  $t$  (if it is a named entity). We identify entity types using the default Lingpipe<sup>2</sup> Named-Entity Recognizer (NER), which recognizes the types *Location*, *Person* and *Organization*. In the above example,  $cp_{v:n}(t)[X]$  would be  $\{\textit{Person}\}$ .

For a rule  $r: LHS \rightarrow RHS$ , we automatically add to  $cp_{v:e}(r)$  all the variable instantiations that were found common for both  $LHS$  and  $RHS$  in a corpus (see Section 4), as in (Pantel et al., 2007; Pennacchiotti et al., 2007). To construct  $cp_{v:n}(r)$ , we currently use a simple approach where each individual term in  $cp_{v:e}(r)$  is analyzed by the NER system, and its type (if any) is added to  $cp_{v:n}(r)$ .

For a template hypothesis, we currently represent  $cp_v(h)$  only by its list of preferred named-entity types,  $cp_{v:n}$ . Similarly to  $cp_g(h)$ , the preferred types for each template variable were adapted from those defined in our IE data (see Section 4).

To allow compatible comparisons with previous work (see Sections 5 and 6), we utilize in this paper only  $cp_{v:e}$  when matching between  $cp_v(r)$  and  $cp_v(t)$ , as only this representation was examined in prior work on context-sensitive rule applications.  $cp_{v:n}$  is utilized for context matches involving  $cp_v(h)$ . We denote the score of matching two  $cp_v$  components by  $m_v(\cdot, \cdot)$ .

<sup>2</sup><http://www.alias-i.com/lingpipe/>

### 3.2.2 Matching $cp_{v:e}$

Our primary matching method is based on replicating the best-performing method reported in (Pantel et al., 2007), which utilizes the CBC distributional word clustering algorithm (Pantel, 2003). In short, this method extends each  $cp_{v:e}$  list with CBC clusters that contain at least one term in the list, scoring them according to their “relevancy”. The score of matching two  $cp_{v:e}$  lists, denoted here  $S_{CBC}(\cdot, \cdot)$ , is the score of the highest scoring member that appears in both lists.

We applied the final binary match score presented in (Pantel et al., 2007), denoted here  $binaryCBC: m_{v:e}(r, t)$  is 1 if  $S_{CBC}(r, t)$  is above a threshold and 0 otherwise. As a more natural ranking method, we also utilize  $S_{CBC}$  directly, denoted  $rankedCBC$ , having  $m_{v:e}(r, t) = S_{CBC}(r, t)$ .

In addition, we tried a simpler method that directly compares the terms in two  $cp_{v:e}$  lists, utilizing the commonly-used term similarity metric of (Lin, 1998a). This method, denoted  $LIN$ , uses the same raw distributional data as CBC but computes only pair-wise similarities, without any clustering phase. We calculated the scores of the 1000 most similar terms for every term in the Reuters RVC1 corpus<sup>3</sup>. Then, a directional similarity of term  $a$  to term  $b$ ,  $s(a, b)$ , is set to be their similarity score if  $a$  is in  $b$ ’s 1000 most similar terms and 0 otherwise. The final score of matching  $r$  with  $t$  is determined by a nearest-neighbor approach, as the score of the most similar pair of terms in the corresponding two lists of the same variable:  $m_{v:e}(r, t) = \max_{j \in vars(r)} [\max_{a \in cp_{v:e}(t)[j], b \in cp_{v:e}(r)[j]} [s(a, b)]]$ .

### 3.2.3 Matching $cp_{v:n}$

We use a simple scoring mechanism for comparing between two named-entity types  $a$  and  $b$ ,  $s(a, b)$ : 1 for identical types and 0.8 otherwise.

A variable  $j$  has a single preferred entity type in  $cp_{v:n}(t)[j]$ , the type of its instantiation in  $t$ . However, it can have several preferred types for  $h$ . When matching  $h$  with  $t$ ,  $j$ ’s match score is that of its highest scoring type, and the final score is the product of all variable scores:  $m_{v:n}(h, t) = \prod_{j \in vars(h)} (\max_{a \in cp_{v:n}(h)[j]} [s(a, cp_{v:n}(t)[j])])$ .

Variable  $j$  may also have several types in  $r$ , the

<sup>3</sup><http://about.reuters.com/researchandstandards/corpus/>

types of the common arguments in  $cp_{v:e}(r)$ . When matching  $h$  with  $r$ ,  $s(a, cp_{v:n}(t)[j])$  is replaced with the average score for  $a$  and each type in  $cp_{v:n}(r)[j]$ .

### 3.3 Overall Score for a Match

A final score for a given match, denoted  $allCP$ , is obtained by the product of all six matching scores of the various CP components (multiplying by 1 if a component score is missing). The six scores are the results of matching any of the two components of  $h$ ,  $t$  and  $r$ :  $m_g(h, t)$ ,  $m_v(h, t)$ ,  $m_g(h, r)$ ,  $m_v(h, r)$ ,  $m_g(r, t)$  and  $m_v(r, t)$  (as specified above,  $m_v(r, t)$  is based on matching  $cp_{v:e}$  while  $m_v(h, r)$  and  $m_v(h, t)$  are based on matching  $cp_{v:n}$ ). We use  $rankedCBC$  for calculating  $m_v(r, t)$ .

Unlike previous work (e.g. (Pantel et al., 2007)), we also utilize the *prior* score of a rule  $r$ , which is provided by the rule-learning algorithm (see next section). We denote by  $allCP+pr$  the final match score obtained by the product of the  $allCP$  score with the prior score of the matched rule.

## 4 Experimental Settings

Evaluating the contribution of Contextual Preferences models requires: (a) a sample of test hypotheses, and (b) a corresponding corpus that contains sentences which entail these hypotheses, where all hypothesis matches (either direct or via rules) are annotated. We found that the available event mention annotations in the ACE 2005 training set<sup>4</sup> provide a useful test set that meets these generic criteria, with the added value of a standard real-world dataset.

The ACE annotation includes 33 types of events, for which all event mentions are annotated in the corpus. The annotation of each mention includes the instantiated arguments for the predicates, which represent the participants in the event, as well as general attributes such as time and place. ACE guidelines specify for each event type its possible arguments, where all arguments are optional. Each argument is associated with a semantic role and a list of possible named-entity types. For instance, an *Injure* event may have the arguments  $\{Agent, Victim, Instrument, Time, Place\}$ , where *Victim* should be a person.

For each event type we manually created a small set of template hypotheses that correspond to the

given event predicate, and specified the appropriate semantic roles for each variable. We considered only binary hypotheses, due to the type of available entailment rules (see below). For *Injure*, the set of hypotheses included ‘*A injure V*’ and ‘*injure V in T*’ where  $role(A)=\{Agent, Instrument\}$ ,  $role(V)=\{Victim\}$ , and  $role(T)=\{Time, Place\}$ . Thus, correct match of an argument corresponds to correct role identification. The templates were represented as Minipar (Lin, 1998b) dependency parse-trees.

The Contextual Preferences for  $h$  were constructed manually: the named-entity types for  $cp_{v:n}(h)$  were set by adapting the entity types given in the guidelines to the types supported by the Lingpipe NER (described in Section 3.2).  $cp_g(h)$  was generated from a short list of nouns and verbs that were extracted from the verbal event definition in the ACE guidelines. For *Injure*, this list included  $\{injure:v, injury:n, wound:v\}$ . This assumes that when writing down an event definition the user would also specify such representative keywords.

Entailment-rules for a given  $h$  (rules in which *RHS* is equal to  $h$ ) were learned automatically by the *DIRT* algorithm (Lin and Pantel, 2001), which also produces a quality score for each rule. We implemented a canonized version of *DIRT* (Szpektor and Dagan, 2007) on the Reuters corpus parsed by Minipar. Each rule’s arguments for  $cp_v(r)$  were also collected from this corpus.

We assessed the CP framework by its ability to correctly rank, for each predicate (event), all the candidate entailing mentions that are found for it in the test corpus. Such ranking evaluation is suitable for unsupervised settings, with a perfect ranking placing all correct mentions before any incorrect ones. The candidate mentions are found in the parsed test corpus by matching the specified event hypotheses, either directly or via the given set of entailment rules, using a syntactic matcher similar to the one in (Szpektor and Dagan, 2007). Finally, the mentions are ranked by their match scores, as described in Section 3.3. As detailed in the next section, those candidate mentions which are also annotated as mentions of the same event in ACE are considered correct.

The evaluation aims to assess the correctness of inferring a target semantic meaning, which is de-

<sup>4</sup><http://projects ldc.upenn.edu/ace/>

noted by a specific predicate. Therefore, we eliminated four ACE event types that correspond to multiple distinct predicates. For instance, the *Transfer-Money* event refers to both *donating* and *lending* money, which are not distinguished by the ACE annotation. We also omitted three events with less than 10 mentions and two events for which the given set of learned rules could not match any mention. We were left with 24 event types for evaluation, which amount to 4085 event mentions in the dataset. Out of these, our binary templates can correctly match only mentions with at least two arguments, which appear 2076 times in the dataset.

Comparing with previous evaluation methodologies, in (Szpektor et al., 2007; Pantel et al., 2007) proper context matching was evaluated by post-hoc judgment of a sample of rule applications for a sample of rules. Such annotation needs to be repeated each time the set of rules is changed. In addition, since the corpus annotation is not exhaustive, recall could not be computed. By contrast, we use a standard real-world dataset, in which all mentions are annotated. This allows immediate comparison of different rule sets and matching methods, without requiring any additional (post-hoc) annotation.

## 5 Results and Analysis

We experimented with three rule setups over the ACE dataset, in order to measure the contribution of the CP framework. In the first setup no rules are used, applying only direct matches of template hypotheses to identify event mentions. In the other two setups we also utilized *DIRT*'s top 50 or 100 rules for each hypothesis.

A match is considered correct when all matched arguments are extracted correctly according to their annotated event roles. This main measurement is denoted *All*. As an additional measurement, denoted *Any*, we consider a match as correct if at least one argument is extracted correctly.

Once event matches are extracted, we first measure for each event its Recall, the number of correct mentions identified out of all annotated event mentions<sup>5</sup> and Precision, the number of correct matches out of all extracted candidate matches. These figures

<sup>5</sup>For Recall, we ignored mentions with less than two arguments, as they cannot be correctly matched by binary templates.

quantify the baseline performance of the *DIRT* rule set used. To assess our ranking quality, we measure for each event the commonly used Average Precision (AP) measure (Voorhees and Harmann, 1998), which is the area under the non-interpolated recall-precision curve, while considering for each setup all correct extracted matches as 100% Recall. Overall, we report *Mean Average Precision (MAP)*, macro average *Precision* and macro average *Recall* over the ACE events. Tables 1 and 2 summarize the main results of our experiments. As far as we know, these are the first published unsupervised results for identifying event arguments in the ACE 2005 dataset.

Examining Recall, we see that it increases substantially when rules are applied: by more than 100% for the top 50 rules, and by about 150% for the top 100, showing the benefit of entailment-rules to covering language variability. The difference between *All* and *Any* results shows that about 65% of the rules that correctly match one argument also match correctly both arguments.

We use two baselines for measuring the CP ranking contribution: Precision, which corresponds to the expected MAP of random ranking, and MAP of ranking using the *prior* rule score provided by *DIRT*. Without rules, the baseline *All* Precision is 34.1%, showing that even the manually constructed hypotheses, which correspond directly to the event predicate, extract event mentions with limited accuracy when context is ignored. When rules are applied, Precision is very low. But ranking is considerably improved using only the prior score (from 1.4% to 22.7% for 50 rules), showing that the prior is an informative indicator for valid matches.

Our main result is that the *allCP* and *allCP+pr* methods rank matches statistically significantly better than the baselines in all setups (according to the Wilcoxon double-sided signed-ranks test at the level of 0.01 (Wilcoxon, 1945)). In the *All* setup, ranking is improved by 70% for direct matching (Table 1). When entailment-rules are also utilized, prior-only ranking is improved by about 35% and 50% when using *allCP* and *allCP+pr*, respectively (Table 2). Figure 2 presents the average Recall-Precision curve of the '50 Rules, All' setup for applying *allCP* or *allCP+pr*, compared to prior-only ranking baseline (other setups behave similarly). The improvement in ranking is evident: the drop in precision is signif-

	R	P	MAP (%)		
	(%)	(%)	$cp_v$	$cp_g$	$allCP$
All	14.0	34.1	46.5	52.2	60.2
Any	21.8	66.0	72.2	80.5	84.1

Table 1: Recall (R), Precision (P) and Mean Average Precision (MAP) when only matching template hypotheses directly.

	#	R	P	MAP (%)		
	Rules	(%)	(%)	prior	$allCP$	$allCP+pr$
All	50	29.6	1.4	22.7	30.6	34.1
	100	34.9	0.7	20.5	26.3	30.2
Any	50	46.5	3.5	41.2	43.7	48.6
	100	52.9	1.8	35.5	35.1	40.8

Table 2: Recall (R), Precision (P) and Mean Average Precision (MAP) when also using rules for matching.

icantly slower when CP is used. The behavior of CP with and without the prior is largely the same up to 50% Recall, but later on our implemented CP models are noisier and should be combined with the prior rule score.

Templates are incorrectly matched for several reasons. First, there are context mismatches which are not scored sufficiently low by our models. Another main cause is incorrect learned rules in which  $LHS$  and  $RHS$  are topically related, e.g. ‘ $X$  convict  $Y \rightarrow X$  arrest  $Y$ ’, or rules that are used in the wrong entailment direction, e.g. ‘ $X$  marry  $Y \rightarrow X$  divorce  $Y$ ’ ( $DIRT$  does not learn rule direction). As such rules do correspond to plausible contexts of the hypothesis, their matches obtain relatively high CP scores. In addition, some incorrect matches are caused by our syntactic matcher, which currently does not handle certain phenomena such as co-reference, modality or negation, and due to Minipar parse errors.

## 5.1 Component Analysis

Table 3 displays the contribution of different CP components to ranking, when adding only that component’s match score to the baselines, and under ablation tests, when using all CP component scores except the tested component, with or without the prior.

As it turns out, matching  $h$  with  $t$  (i.e.  $cp(h, t)$ , which combines  $cp_g(h, t)$  and  $cp_v(h, t)$ ) is most useful. With our current models, using only  $cp(h, t)$  along with the prior, while ignoring  $cp(r)$ , achieves

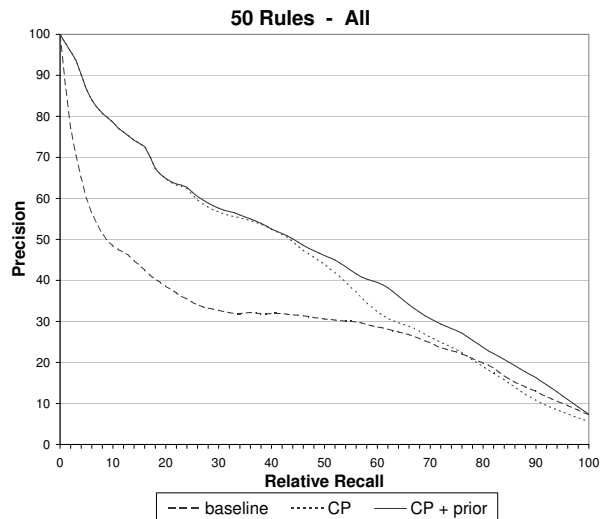


Figure 2: Recall-Precision curves for ranking using: (a) only the prior (baseline); (b)  $allCP$ ; (c)  $allCP+pr$ .

the highest score in the table. The strong impact of matching  $h$  and  $t$ ’s preferences is also evident in Table 1, where ranking based on either  $cp_g$  or  $cp_v$  substantially improves precision, while their combination provides the best ranking. These results indicate that the two CP components capture complementary information and both are needed to assess the correctness of a match.

When ignoring the prior rule score,  $cp(r, t)$  is the major contributor over the baseline Precision. For  $cp_v(r, t)$ , this is in synch with the result in (Pantel et al., 2007), which is based on this single model without utilizing prior rule scores. On the other hand,  $cp_v(r, t)$  does not improve the ranking when the prior is used, suggesting that this contextual model for the rule’s variables is not stronger than the context-insensitive prior rule score. Furthermore, relative to this  $cp_v(r, t)$  model from (Pantel et al., 2007), our combined  $allCP$  model, with or without the prior (first row of Table 2), obtains statistically significantly better ranking (at the level of 0.01).

Comparing between the algorithms for matching  $cp_{v:e}$  (Section 3.2.2) we found that while  $rankedCBC$  is statistically significantly better than  $binaryCBC$ ,  $rankedCBC$  and  $LIN$  generally achieve the same results. When considering the tradeoffs between the two,  $LIN$  is based on a much simpler learning algorithm while  $CBC$ ’s output is more compact and allows faster CP matches.

	Addition To		Ablation From	
	P	prior	<i>allCP</i>	<i>allCP+pr</i>
Baseline	1.4	22.7	30.6	34.1
$cp_g(h, t)$	*10.4	*35.4	32.4	33.7
$cp_v(h, t)$	*11.0	29.9	27.6	32.9
$cp(h, t)$	*8.9	<b>*37.5</b>	28.6	30.0
$cp_g(r, t)$	*4.2	*30.6	<b>32.5</b>	35.4
$cp_v(r, t)$	*21.7	21.9	*12.9	33.6
$cp(r, t)$	*26.0	*29.6	*17.9	<b>36.8</b>
$cp_g(h, r)$	*8.1	22.4	31.9	34.3
$cp_v(h, r)$	*10.7	22.7	*27.9	34.4
$cp(h, r)$	*16.5	22.4	*29.2	34.4
$cp_g(h, r, t)$	*7.7	*30.2	*27.5	*29.2
$cp_v(h, r, t)$	<b>*27.5</b>	29.2	*7.7	30.2

\* Indicates statistically significant changes compared to the baseline, according to the Wilcoxon test at the level of 0.01.

Table 3: MAP(%), under the ‘50 rules, All’ setup, when adding component match scores to Precision (P) or prior-only MAP baselines, and when ranking with *allCP* or *allCP+pr* methods but ignoring that component scores.

Currently, some models do not improve the results when the prior is used. Yet, we would like to further weaken the dependency on the prior score, since it is biased towards frequent contexts. We aim to properly identify also infrequent contexts (or meanings) at inference time, which may be achieved by better CP models. More generally, when used on top of all other components, some of the models slightly degrade performance, as can be seen by those figures in the ablation tests which are higher than the corresponding baseline. However, due to their different roles, each of the matching components might capture some unique preferences. For example,  $cp(h, r)$  should be useful to filter out rules that don’t match the intended meaning of the given  $h$ . Overall, this suggests that future research for better models should aim to obtain a marginal improvement by each component.

## 6 Related Work

Context sensitive inference was mainly investigated in an application-dependent manner. For example, (Harabagiu et al., 2003) describe techniques for identifying the question focus and the answer type in QA. (Patwardhan and Riloff, 2007) propose a supervised approach for IE, in which relevant text regions

for a target relation are identified prior to applying extraction rules.

Recently, the need for context-aware inference was raised (Szpektor et al., 2007). (Pantel et al., 2007) propose to learn the preferred instantiations of rule variables, termed Inferential Selectional Preferences (ISP). Their clustering-based model is the one we implemented for  $m_v(r, t)$ . A similar approach is taken in (Pennacchiotti et al., 2007), where LSA similarity is used to compare between the preferred variable instantiations for a rule and their instantiations in the matched text. (Downey et al., 2007) use HMM-based similarity for the same purpose. All these methods are analogous to matching  $cp_v(r)$  with  $cp_v(t)$  in the CP framework.

(Dagan et al., 2006; Connor and Roth, 2007) proposed generic approaches for identifying valid applications of lexical rules by classifying the surrounding global context of a word as valid or not for that rule. These approaches are analogous to matching  $cp_g(r)$  with  $cp_g(t)$  in our framework.

## 7 Conclusions

We presented the Contextual Preferences (CP) framework for assessing the validity of inferences in context. CP enriches the representation of textual objects with typical contextual information that constrains or disambiguates their meaning, and provides matching functions that compare the preferences of objects involved in the inference. Experiments with our implemented CP models, over real-world IE data, show significant improvements relative to baselines and some previous work.

In future research we plan to investigate improved models for representing and matching CP, and to extend the experiments to additional applied datasets. We also plan to apply the framework to lexical inference rules, for which it seems directly applicable.

## Acknowledgements

The authors would like to thank Alfio Massimiliano Gliozzo for valuable discussions. This work was partially supported by ISF grant 1095/05, the IST Programme of the European Community under the PASCAL Network of Excellence IST-2002-506778, the NEGEV project ([www.negev-initiative.org](http://www.negev-initiative.org)) and the FBK-irst/Bar-Ilan University collaboration.

## References

- Roy Bar-Haim, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*.
- Michael Connor and Dan Roth. 2007. Context sensitive paraphrasing with a global unsupervised classifier. In *Proceedings of the European Conference on Machine Learning (ECML)*.
- Ido Dagan, Oren Glickman, Alfio Gliozzo, Efrat Marmorstein, and Carlo Strapparava. 2006. Direct word sense matching for lexical substitution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of ACL*.
- Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. 2005. An inference model for semantic entailment in natural language. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.
- Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.
- Doug Downey, Stefan Schoenmackers, and Oren Etzioni. 2007. Sparse information extraction: Unsupervised language models to the rescue. In *Proceedings of the 45th Annual Meeting of ACL*.
- Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.
- Alfio Massimiliano Gliozzo. 2005. *Semantic Domains in Computational Linguistics*. Ph.D. thesis. Advisor-Carlo Strapparava.
- Sanda M. Harabagiu, Steven J. Maiorano, and Marius A. Paşca. 2003. Open-domain textual question answering techniques. *Nat. Lang. Eng.*, 9(3):231–267.
- Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. In *Natural Language Engineering*, volume 7(4), pages 343–360.
- Dekang Lin. 1998a. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*.
- Dekang Lin. 1998b. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC*.
- Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. 2000. The structure and performance of an open-domain question answering system. In *Proceedings of the 38th Annual Meeting of ACL*.
- Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *Human Language Technologies 2007: The Conference of NAACL; Proceedings of the Main Conference*.
- Patrick Andre Pantel. 2003. *Clustering by committee*. Ph.D. thesis. Advisor-Dekang Lin.
- Siddharth Patwardhan and Ellen Riloff. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.
- Marco Pennacchiotti, Roberto Basili, Diego De Cao, and Paolo Marocco. 2007. Learning selectional preferences for entailment or paraphrasing rules. In *Proceedings of RANLP*.
- Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of ACL*.
- Lorenza Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan, and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of the 11th Conference of the EACL*.
- Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between ne pairs. In *Proceedings of IWP*.
- Yusuke Shinyama, Satoshi Sekine, Sudo Kiyoshi, and Ralph Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of Human Language Technology Conference*.
- Idan Szpektor and Ido Dagan. 2007. Learning canonical forms of entailment rules. In *Proceedings of RANLP*.
- Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP 2004*, pages 41–48, Barcelona, Spain.
- Idan Szpektor, Eyal Shnarch, and Ido Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *Proceedings of the 45th Annual Meeting of ACL*.
- Ellen M. Voorhees and Donna Harman. 1998. Overview of the seventh text retrieval conference (trec-7). In *The Seventh Text Retrieval Conference*.
- Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

# Unsupervised Discovery of Generic Relationships Using Pattern Clusters and its Evaluation by Automatically Generated SAT Analogy Questions

**Dmitry Davidov**

ICNC

Hebrew University of Jerusalem

dmitry@alice.nc.huji.ac.il

**Ari Rappoport**

Institute of Computer Science

Hebrew University of Jerusalem

arir@cs.huji.ac.il

## Abstract

We present a novel framework for the discovery and representation of general semantic relationships that hold between lexical items. We propose that each such relationship can be identified with a cluster of patterns that captures this relationship. We give a fully unsupervised algorithm for pattern cluster discovery, which searches, clusters and merges high-frequency words-based patterns around randomly selected hook words. Pattern clusters can be used to extract instances of the corresponding relationships. To assess the quality of discovered relationships, we use the pattern clusters to automatically generate SAT analogy questions. We also compare to a set of known relationships, achieving very good results in both methods. The evaluation (done in both English and Russian) substantiates the premise that our pattern clusters indeed reflect relationships perceived by humans.

## 1 Introduction

Semantic resources can be very useful in many NLP tasks. Manual construction of such resources is labor intensive and susceptible to arbitrary human decisions. In addition, manually constructed semantic databases are not easily portable across text domains or languages. Hence, there is a need for developing semantic acquisition algorithms that are as unsupervised and language independent as possible.

A fundamental type of semantic resource is that of concepts (represented by sets of lexical items) and their inter-relationships. While there is relatively good agreement as to what concepts are

and which concepts should exist in a lexical resource, identifying types of important lexical relationships is a rather difficult task. Most established resources (e.g., WordNet) represent only the main and widely accepted relationships such as hypernymy and meronymy. However, there are many other useful relationships between concepts, such as noun-modifier and inter-verb relationships. Identifying and representing these explicitly can greatly assist various tasks and applications. There are already applications that utilize such knowledge (e.g., (Tatu and Moldovan, 2005) for textual entailment).

One of the leading methods in semantics acquisition is based on patterns (see e.g., (Hearst, 1992; Pantel and Pennacchiotti, 2006)). The standard process for pattern-based relation extraction is to start with hand-selected patterns or word pairs expressing a particular relationship, and iteratively scan the corpus for co-appearances of word pairs in patterns and for patterns that contain known word pairs. This methodology is semi-supervised, requiring pre-specification of the desired relationship or hand-coding initial seed words or patterns. The method is quite successful, and examining its results in detail shows that concept relationships are often being manifested by several different patterns.

In this paper, unlike the majority of studies that use patterns in order to find instances of given relationships, we use sets of patterns as the *definitions* of lexical relationships. We introduce *pattern clusters*, a novel framework in which each cluster corresponds to a relationship that can hold between the lexical items that fill its patterns' slots. We present a fully unsupervised algorithm to compute pat-



tern clusters, not requiring any, even implicit, pre-specification of relationship types or word/pattern seeds. Our algorithm does not utilize preprocessing such as POS tagging and parsing. Some patterns may be present in several clusters, thus indirectly addressing pattern ambiguity.

The algorithm is comprised of the following stages. First, we randomly select hook words and create a context corpus (hook corpus) for each hook word. Second, we define a meta-pattern using high frequency words and punctuation. Third, in each hook corpus, we use the meta-pattern to discover concrete patterns and target words co-appearing with the hook word. Fourth, we cluster the patterns in each corpus according to co-appearance of the target words. Finally, we merge clusters from different hook corpora to produce the final structure. We also propose a way to label each cluster by word pairs that represent it best.

Since we are dealing with relationships that are unspecified in advance, assessing the quality of the resulting pattern clusters is non-trivial. Our evaluation uses two methods: SAT tests, and comparison to known relationships. We used instances of the discovered relationships to automatically generate analogy SAT tests in two languages, English and Russian<sup>1</sup>. Human subjects answered these and real SAT tests. English grades were 80% for our test and 71% for the real test (83% and 79% for Russian), showing that our relationship definitions indeed reflect human notions of relationship similarity. In addition, we show that among our pattern clusters there are clusters that cover major known noun-compound and verb-verb relationships.

In the present paper we focus on the pattern cluster resource itself and how to evaluate its intrinsic quality. In (Davidov and Rappoport, 2008) we show how to *use* the resource for a known task of a totally different nature, classification of relationships between nominals (based on annotated data), obtaining superior results over previous work.

Section 2 discusses related work, and Section 3 presents the pattern clustering and labeling algorithm. Section 4 describes the corpora we used and the algorithm's parameters in detail. Sections 5 and

---

<sup>1</sup>Turney and Littman (2005) automatically answers SAT tests, while our focus is on generating them.

6 present SAT and comparison evaluation results.

## 2 Related Work

Extraction of relation information from text is a large sub-field in NLP. Major differences between pattern approaches include the relationship types sought (including domain restrictions), the degrees of supervision and required preprocessing, and evaluation method.

### 2.1 Relationship Types

There is a large body of related work that deals with discovery of basic relationship types represented in useful resources such as WordNet, including hypernymy (Hearst, 1992; Pantel et al., 2004; Snow et al., 2006), synonymy (Davidov and Rappoport, 2006; Widdows and Dorow, 2002) and meronymy (Berland and Charniak, 1999; Girju et al., 2006).

Since named entities are very important in NLP, many studies define and discover relations between named entities (Hasegawa et al., 2004; Hassan et al., 2006). Work was also done on relations between verbs (Chklovski and Pantel, 2004). There is growing research on relations between nominals (Moldovan et al., 2004; Girju et al., 2007).

### 2.2 Degree of Supervision and Preprocessing

While numerous studies attempt to discover one or more pre-specified relationship types, very little previous work has directly attempted the discovery of which main types of generic relationships actually exist in an unrestricted domain. Turney (2006) provided a pattern distance measure that allows a fully unsupervised measurement of relational similarity between two pairs of words; such a measure could in principle be used by a clustering algorithm in order to deduce relationship types, but this was not discussed. Unlike (Turney, 2006), we do not perform any pattern ranking. Instead we produce (possibly overlapping) hard clusters, where each pattern cluster represents a relationship discovered in the domain. Banko et al. (2007) and Rosenfeld and Feldman (2007) find relationship instances where the relationships are not specified in advance. They aim to find relationship instances rather than identify generic semantic relationships. Thus, their representation is very different from ours. In addition, (Banko et al., 2007) utilize supervised tools such

as a POS tagger and a shallow parser. Davidov et al. (2007) proposed a method for unsupervised discovery of concept-specific relations. That work, like ours, relies on pattern clusters. However, it requires initial word seeds and targets the discovery of relationships specific for some given concept, while we attempt to discover and define generic relationships that exist in the entire domain.

Studying relationships between tagged named entities, (Hasegawa et al., 2004; Hassan et al., 2006) proposed unsupervised clustering methods that assign given sets of pairs into several clusters, where each cluster corresponds to one of a known set of relationship types. Their classification setting is thus very different from our unsupervised discovery one.

Several recent papers discovered relations on the web using seed patterns (Pantel et al., 2004), rules (Etzioni et al., 2004), and word pairs (Pasca et al., 2006; Alfonseca et al., 2006). The latter used the notion of hook which we also use in this paper. Several studies utilize some preprocessing, including parsing (Hasegawa et al., 2004; Hassan et al., 2006) and usage of syntactic (Suchanek et al., 2006) and morphological (Pantel et al., 2004) information in patterns. Several algorithms use manually-prepared resources, including WordNet (Moldovan et al., 2004; Costello et al., 2006) and Wikipedia (Strube and Ponzetto, 2006). In this paper, we do not utilize any language-specific preprocessing or any other resources, which makes our algorithm relatively easily portable between languages, as we demonstrate in our bilingual evaluation.

### 2.3 Evaluation Method

Evaluation for hypernymy and synonymy usually uses WordNet (Lin and Pantel, 2002; Widdows and Dorow, 2002; Davidov and Rappoport, 2006). For more specific lexical relationships like relationships between verbs (Chklovski and Pantel, 2004), nominals (Girju et al., 2004; Girju et al., 2007) or meronymy subtypes (Berland and Charniak, 1999) there is still little agreement which important relationships should be defined. Thus, there are more than a dozen different type hierarchies and tasks proposed for noun compounds (and nominals in general), including (Nastase and Szpakowicz, 2003; Girju et al., 2005; Girju et al., 2007).

There are thus two possible ways for a fair eval-

uation. A study can develop its own relationship definitions and dataset, like (Nastase and Szpakowicz, 2003), thus introducing a possible bias; or it can accept the definition and dataset prepared by another work, like (Turney, 2006). However, this makes it impossible to work on new relationship types. Hence, when exploring very specific relationship types or very generic, but not widely accepted, types (like verb strength), many researchers resort to manual human-based evaluation (Chklovski and Pantel, 2004). In our case, where relationship types are not specified in advance, creating an unbiased benchmark is very problematic, so we rely on human subjects for relationship evaluation.

## 3 Pattern Clustering Algorithm

Our algorithm first discovers and clusters patterns in which a single ('hook') word participates, and then merges the resulting clusters to form the final structure. In this section we detail the algorithm. The algorithm utilizes several parameters, whose selection is detailed in Section 4. We refer to a pattern contained in our clusters (a pattern type) as a 'pattern' and to an occurrence of a pattern in the corpus (a pattern token) as a 'pattern instance'.

### 3.1 Hook Words and Hook Corpora

As a first step, we randomly select a set of hook words. Hook words were used in e.g. (Alfonseca et al., 2006) for extracting general relations starting from given seed word pairs. Unlike most previous work, our hook words are not provided in advance but selected randomly; the goal in those papers is to discover relationships between given word pairs, while we use hook words in order to discover relationships that generally occur in the corpus.

Only patterns in which a hook word actually participates will eventually be discovered. Hence, in principle we should select as many hook words as possible. However, words whose frequency is very high are usually ambiguous and are likely to produce patterns that are too noisy, so we do not select words with frequency higher than a parameter  $F_C$ . In addition, we do not select words whose frequency is below a threshold  $F_B$ , to avoid selection of typos and other noise that frequently appear on the web.

We also limit the total number  $N$  of hook words.

Our algorithm merges clusters originating from different hook words. Using too many hook words increases the chance that some of them belong to a noisy part in the corpus and thus lowers the quality of our resulting clusters.

For each hook word, we now create a hook corpus, the set of the contexts in which the word appears. Each context is a window containing  $W$  words or punctuation characters before and after the hook word. We avoid extracting text from clearly unformatted sentences and our contexts do not cross paragraph boundaries.

The size of each hook corpus is much smaller than that of the whole corpus, easily fitting into main memory; the corpus of a hook word occurring  $h$  times in the corpus contains at most  $2hW$  words. Since most operations are done on each hook corpus separately, computation is very efficient.

Note that such context corpora can in principle be extracted by focused querying on the web, making the system dynamically scalable. It is also possible to restrict selection of hook words to a specific domain or word type, if we want to discover only a desired subset of existing relationships. Thus we could sample hook words from nouns, verbs, proper names, or names of chemical compounds if we are only interested in discovering relationships between these. Selecting hook words randomly allows us to avoid using any language-specific data at this step.

### 3.2 Pattern Specification

In order to reduce noise and to make the computation more efficient, we did not consider all contexts of a hook word as pattern candidates, only contexts that are instances of a specified meta-pattern type. Following (Davidov and Rappoport, 2006), we classified words into high-frequency words (HFWs) and content words (CWs). A word whose frequency is more (less) than  $F_H$  ( $F_C$ ) is considered to be a HFW (CW). Unlike (Davidov and Rappoport, 2006), we consider all punctuation characters as HFWs. Our patterns have the general form

**[Prefix]**  $CW_1$  **[Infix]**  $CW_2$  **[Postfix]**

where Prefix, Infix and Postfix contain only HFWs. To reduce the chance of catching  $CW_i$ 's that are parts of a multiword expression, we require Prefix and Postfix to have at least one word (HFW), while

Infix is allowed to contain any number of HFWs (but recall that the total length of a pattern is limited by window size). A pattern example is '*such X as Y and*'. During this stage we only allow single words to be in CW slots<sup>2</sup>.

### 3.3 Discovery of Target Words

For each of the hook corpora, we now extract all pattern instances where one CW slot contains the hook word and the other CW slot contains some other ('target') word. To avoid the selection of common words as target words, and to avoid targets appearing in pattern instances that are relatively fixed multiword expressions, we sort all target words in a given hook corpus by pointwise mutual information between hook and target, and drop patterns obtained from pattern instances containing the lowest and highest  $L$  percent of target words.

### 3.4 Local Pattern Clustering

We now have for each hook corpus a set of patterns. All of the corresponding pattern instances share the hook word, and some of them also share a target word. We cluster patterns in a two-stage process. First, we group in clusters all patterns whose instances share the same target word, and ignore the rest. For each target word we have a single pattern cluster. Second, we merge clusters that share more than  $S$  percent of their patterns. A pattern can appear in more than a single cluster. Note that clusters contain pattern *types*, obtained through examining pattern *instances*.

### 3.5 Global Cluster Merging

The purpose of this stage is to create clusters of patterns that express generic relationships rather than ones specific to a single hook word. In addition, the technique used in this stage reduces noise. For each created cluster we will define *core* patterns and *unconfirmed* patterns, which are weighed differently during cluster labeling (see Section 3.6). We merge clusters from different hook corpora using the following algorithm:

1. Remove all patterns originating from a single hook corpus.

<sup>2</sup>While for pattern clusters creation we use only single words as CWs, later during evaluation we allow multiword expressions in CW slots of previously acquired patterns.

2. Mark all patterns of all present clusters as unconfirmed.
3. While there exists some cluster  $C_1$  from corpus  $D_X$  containing only unconfirmed patterns:
  - (a) Select a cluster with a minimal number of patterns.
  - (b) For each corpus  $D$  different from  $D_X$ :
    - i. Scan  $D$  for clusters  $C_2$  that share at least  $S$  percent of their patterns, and all of their core patterns, with  $C_1$ .
    - ii. Add all patterns of  $C_2$  to  $C_1$ , setting all shared patterns as core and all others as unconfirmed.
    - iii. Remove cluster  $C_2$ .
  - (c) If all of  $C_1$ 's patterns remain unconfirmed remove  $C_1$ .
4. If several clusters have the same set of core patterns merge them according to rules (i,ii).

We start from the smallest clusters because we expect these to be more precise; the best patterns for semantic acquisition are those that belong to small clusters, and appear in many different clusters. At the end of this algorithm, we have a set of pattern clusters where for each cluster there are two subsets, core patterns and unconfirmed patterns.

### 3.6 Labeling of Pattern Clusters

To label pattern clusters we define a HITS measure that reflects the affinity of a given word pair to a given cluster. For a given word pair  $(w_1, w_2)$  and cluster  $C$  with  $n$  core patterns  $P_{core}$  and  $m$  unconfirmed patterns  $P_{unconf}$ ,

$$Hits(C, (w_1, w_2)) = \frac{|\{p; (w_1, w_2) \text{ appears in } p \in P_{core}\}|}{n} + \alpha \times \frac{|\{p; (w_1, w_2) \text{ appears in } p \in P_{unconf}\}|}{m}.$$

In this formula, ‘appears in’ means that the word pair appears in instances of this pattern extracted from the original corpus or retrieved from the web during evaluation (see Section 5.2). Thus if some pair appears in most of patterns of some cluster it receives a high HITS value for this cluster. The top 5 pairs for each cluster are selected as its labels.  $\alpha \in (0..1)$  is a parameter that lets us modify the relative weight of core and unconfirmed patterns.

## 4 Corpora and Parameters

In this section we describe our experimental setup, and discuss in detail the effect of each of the algorithms’ parameters.

### 4.1 Languages and Corpora

The evaluation was done using corpora in English and Russian. The English corpus (Gabrilovich and Markovitch, 2005) was obtained through crawling the URLs in the Open Directory Project (dmoz.org). It contains about 8.2G words and its size is about 68GB of untagged plain text. The Russian corpus was collected over the web, comprising a variety of domains, including news, web pages, forums, novels and scientific papers. It contains 7.5G words of size 55GB untagged plain text. Aside from removing noise and sentence duplicates, we did not apply any text preprocessing or tagging.

### 4.2 Parameters

Our algorithm uses the following parameters:  $F_C$ ,  $F_H$ ,  $F_B$ ,  $W$ ,  $N$ ,  $L$ ,  $S$  and  $\alpha$ . We used part of the Russian corpus as a development set for determining the parameters. On our development set we have tested various parameter settings. A detailed analysis of the involved parameters is beyond the scope of this paper; below we briefly discuss the observed qualitative effects of parameter selection. Naturally, the parameters are not mutually independent.

$F_C$  (upper bound for content word frequency in patterns) influences which words are considered as hook and target words. More ambiguous words generally have higher frequency. Since content words determine the joining of patterns into clusters, the more ambiguous a word is, the noisier the resulting clusters. Thus, higher values of  $F_C$  allow more ambiguous words, increasing cluster recall but also increasing cluster noise, while lower ones increase cluster precision at the expense of recall.

$F_H$  (lower bound for HFW frequency in patterns) influences the specificity of patterns. Higher values restrict our patterns to be based upon the few most common HFWs (like ‘the’, ‘of’, ‘and’) and thus yield patterns that are very generic. Lowering the values, we obtain increasing amounts of pattern clusters for more specific relationships. The value we use for  $F_H$  is lower than that used for  $F_C$ , in order to allow as HFWs function words of relatively low frequency (e.g., ‘through’), while allowing as content words some frequent words that participate in meaningful relationships (e.g., ‘game’). However, this way we may also introduce more noise.

$F_B$  (lower bound for hook words) filters hook words that do not appear enough times in the corpus. We have found that this parameter is essential for removing typos and other words that do not qualify as hook words.

$N$  (number of hook words) influences relationship coverage. With higher  $N$  values we discover more relationships roughly of the same specificity level, but computation becomes less efficient and more noise is introduced.

$W$  (window size) determines the length of the discovered patterns. Lower values are more efficient computationally, but values that are too low result in drastic decrease in coverage. Higher values would be more useful when we allow our algorithm to support multiword expressions as hooks and targets.

$L$  (target word mutual information filter) helps in avoiding using as targets common words that are unrelated to hooks, while still catching as targets frequent words that are related. Low  $L$  values decrease pattern precision, allowing patterns like ‘give  $X$  please  $Y$  more’, where  $X$  is the hook (e.g., ‘Alex’) and  $Y$  the target (e.g., ‘some’). High values increase pattern precision at the expense of recall.

$S$  (minimal overlap for cluster merging) is a clusters merge filter. Higher values cause more strict merging, producing smaller but more precise clusters, while lower values start introducing noise. In extreme cases, low values can start a chain reaction of total merging.

$\alpha$  (core vs. unconfirmed weight for HITS labeling) allows lower quality patterns to complement higher quality ones during labeling. Higher values increase label noise, while lower ones effectively ignore unconfirmed patterns during labeling.

In our experiments we have used the following values (again, determined using a development set) for these parameters:  $F_C$ : 1,000 words per million (wpm);  $F_H$ : 100 wpm;  $F_B$ : 1.2 wpm;  $N$ : 500 words;  $W$ : 5 words;  $L$ : 30%;  $S$ : 2/3;  $\alpha$ : 0.1.

## 5 SAT-based Evaluation

As discussed in Section 2, the evaluation of semantic relationship structures is non-trivial. The goal of our evaluation was to assess whether pattern clusters indeed represent meaningful, precise and different relationships. There are two complementary perspec-

tives that a pattern clusters quality assessment needs to address. The first is the quality (precision/recall) of individual pattern clusters: does each pattern cluster capture lexical item pairs of the same semantic relationship? does it recognize many pairs of the same semantic relationship? The second is the quality of the cluster set as whole: does the pattern clusters set allow identification of important known semantic relationships? do several pattern clusters describe the same relationship?

Manually examining the resulting pattern clusters, we saw that the majority of sampled clusters indeed clearly express an interesting specific relationship. Examples include familiar hypernymy clusters such as<sup>3</sup> {‘such  $X$  as  $Y$ ’, ‘ $X$  such as  $Y$ ’, ‘ $Y$  and other  $X$ ’,} with label (*pets, dogs*), and much more specific clusters like {‘buy  $Y$  accessory for  $X$ !’, ‘shipping  $Y$  for  $X$ ’, ‘ $Y$  is available for  $X$ ’, ‘ $Y$  are available for  $X$ ’, ‘ $Y$  are available for  $X$  systems’, ‘ $Y$  for  $X$ ’}, labeled by (*phone, charger*). Some clusters contain overlapping patterns, like ‘ $Y$  for  $X$ ’, but represent different relationships when examined as a whole.

We addressed the evaluation questions above using a SAT-like analogy test automatically generated from word pairs captured by our clusters (see below in this section). In addition, we tested coverage and overlap of pattern clusters with a set of 35 known relationships, and we compared our patterns to those found useful by other algorithms (the next section).

Quantitatively, the final number of clusters is 508 (470) for English (Russian), and the average cluster size is 5.5 (6.1) pattern types. 55% of the clusters had no overlap with other clusters.

### 5.1 SAT Analogy Choice Test

Our main evaluation method, which is also a useful application by itself, uses our pattern clusters to automatically generate SAT analogy questions. The questions were answered by human subjects.

We randomly selected 15 clusters. This allowed us to assess the precision of the whole cluster set as well as of the internal coherence of separate clusters (see below). For each cluster, we constructed a SAT analogy question in the following manner. The header of the question is a word pair that is one of the label pairs of the cluster. The five multiple

<sup>3</sup>For readability, we omit punctuations in Prefix and Postfix.

choice items include: (1) another label of the cluster (the ‘correct’ answer); (2) three labels of other clusters among the 15; and (3) a pair constructed by randomly selecting words from those making up the various cluster labels.

In our sample there were no word pairs assigned as labels to more than one cluster<sup>4</sup>. As a baseline for comparison, we have mixed these questions with 15 real SAT questions taken from English and Russian SAT analogy tests. In addition, we have also asked our subjects to write down one example pair of the same relationship for each question in the test.

As an example, from one of the 15 clusters we have randomly selected the label (*glass, water*). The correct answer selected from the same cluster was (*schoolbag, book*). The three pairs randomly selected from the other 14 clusters were (*war, death*), (*request, license*) and (*mouse, cat*). The pair randomly selected from a cluster not among the 15 clusters was (*milk, drink*). Among the subjects’ proposals for this question were (*closet, clothes*) and (*wallet, money*).

We computed accuracy of SAT answers, and the correlation between answers for our questions and the real ones (Table 1). Three things are demonstrated about our system when humans are capable of selecting the correct answer. First, our clusters are internally coherent in the sense of expressing a certain relationship, because people identified that the pairs in the question header and in the correct answer exhibit the same relationship. Second, our clusters distinguish between different relationships, because the three pairs not expressing the same relationship as the header were not selected by the evaluators. Third, our cluster labeling algorithm produces results that are usable by people.

The test was performed in both English and Russian, with 10 (6) subjects for English (Russian). The subjects (biology and CS students) were not involved with the research, did not see the clusters, and did not receive any special training as preparation. Inter-subject agreement and Kappa were 0.82, 0.72 (0.9, 0.78) for English (Russian). As reported in (Turney, 2005), an average high-school SAT grade is 57. Table 1 shows the final English and Rus-

<sup>4</sup>But note that a pair can certainly obtain a positive HITS value for several clusters.

	Our method	Real SAT	Correlation
English	<b>80%</b>	71%	0.85
Russian	<b>83%</b>	79%	0.88

Table 1: Pattern cluster evaluation using automatically generated SAT analogy choice questions.

sian grade average for ours and real SAT questions.

We can see that for both languages, around 80% of the choices were correct (the random choice baseline is 20%). Our subjects are university students, so results higher than 57 are expected, as we can see from real SAT performance. The difference in grades between the two languages might be attributed to the presence of relatively hard and uncommon words. It also may result from the Russian test being easier because there is less verb-noun ambiguity in Russian.

We have observed a high correlation between true grades and ours, suggesting that our automatically generated test reflects the ability to recognize analogies and can be potentially used for automated generation of SAT-like tests.

The results show that our pattern clusters indeed mirror a human notion of relationship similarity and represent meaningful relationships. They also show that as intended, different clusters describe different relationships.

## 5.2 Analogy Invention Test

To assess recall of separate pattern clusters, we have asked subjects to provide (if possible) an additional pair for each SAT question. On each such pair we have automatically extracted a set of pattern instances that capture this pair by using automated web queries. Then we calculated the HITS value for each of the selected pairs and assigned them to clusters with highest HITS value. The numbers of pairs provided were 81 for English and 43 for Russian.

We have estimated precision for this task as macro-average of percentage of correctly assigned pairs, obtaining 87% for English and 82% for Russian (the random baseline of this 15-class classification task is 6.7%). It should be noted however that the human-provided additional relationship examples in this test are not random so it may introduce bias. Nevertheless, these results confirm that our pattern clusters are able to recognize new in-

30 Noun Compound Relationships		
	Avg. num of clusters	Overlap
Russian	1.8	0.046
English	1.7	0.059
5 Verb Verb Relationships		
Russian	1.4	0.01
English	1.2	0

Table 2: Patterns clusters discovery of known relationships.

stances of relationships of the same type.

## 6 Evaluation Using Known Information

We also evaluated our pattern clusters using relevant information reported in related work.

### 6.1 Discovery of Known Relationships

To estimate recall of our pattern cluster set, we attempted to estimate whether (at least) a subset of known relationships have corresponding pattern clusters. As a testing subset, we have used 35 relationships for both English and Russian. 30 relations are noun compound relationships as proposed in the (Nastase and Szpakowicz, 2003) classification scheme, and 5 relations are verb-verb relations proposed by (Chklovski and Pantel, 2004). We have manually created sets of 5 unambiguous sample pairs for each of these 35 relationships. For each such pair we have assigned the pattern cluster with best HITS value.

The middle column of Table 2 shows the average number of clusters per relationship. Ideally, if for each relationship all 5 pairs are assigned to the same cluster, the average would be 1. In the worst case, when each pair is assigned to a different cluster, the average would be 5. We can see that most of the pairs indeed fall into one or two clusters, successfully recognizing that similarly related pairs belong to the same cluster. The column on the right shows the overlap between different clusters, measured as the average number of shared pairs in two randomly selected clusters. The baseline in this case is essentially 5, since there are more than 400 clusters for 5 word pairs. We see a very low overlap between assigned clusters, which shows that these clusters indeed separate well between defined relations.

### 6.2 Discovery of Known Pattern Sets

We compared our clusters to lists of patterns reported as useful by previous papers. These lists included patterns expressing hypernymy (Hearst, 1992; Pantel et al., 2004), meronymy (Berland and Charniak, 1999; Girju et al., 2006), synonymy (Widdows and Dorow, 2002; Davidov and Rapoport, 2006), and verb strength + verb happens-before (Chklovski and Pantel, 2004). In all cases, we discovered clusters containing all of the reported patterns (including their refinements with domain-specific prefix or postfix) and not containing patterns of competing relationships.

## 7 Conclusion

We have proposed a novel way to define and identify generic lexical relationships as clusters of patterns. Each such cluster is set of patterns that can be used to identify, classify or capture new instances of some unspecified semantic relationship. We showed how such pattern clusters can be obtained automatically from text corpora without any seeds and without relying on manually created databases or language-specific text preprocessing. In an evaluation based on an automatically created analogy SAT test we showed on two languages that pairs produced by our clusters indeed strongly reflect human notions of relation similarity. We also showed that the obtained pattern clusters can be used to recognize new examples of the same relationships. In an additional test where we assign labeled pairs to pattern clusters, we showed that they provide good coverage for known noun-noun and verb-verb relationships for both tested languages.

While our algorithm shows good performance, there is still room for improvement. It utilizes a set of constants that affect precision, recall and the granularity of the extracted cluster set. It would be beneficial to obtain such parameters automatically and to create a multilevel relationship hierarchy instead of a flat one, thus combining different granularity levels. In this study we applied our algorithm to a generic domain, while the same method can be used for more restricted domains, potentially discovering useful domain-specific relationships.

## References

- Alfonseca, E., Ruiz-Casado, M., Okumura, M., Castells, P., 2006. Towards large-scale non-taxonomic relation extraction: estimating the precision of rote extractors. *COLING-ACL '06 Ontology Learning & Population Workshop*.
- Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O., 2007. Open information extraction from the Web. *IJCAI '07*.
- Berland, M., Charniak, E., 1999. Finding parts in very large corpora. *ACL '99*.
- Chklovski, T., Pantel, P., 2004. VerbOcean: mining the web for fine-grained semantic verb relations. *EMNLP '04*.
- Costello, F., Veale, T. Dunne, S., 2006. Using WordNet to automatically deduce relations between words in noun-noun compounds. *COLING-ACL '06*.
- Davidov, D., Rappoport, A., 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. *COLING-ACL '06*.
- Davidov, D., Rappoport, A. and Koppel, M., 2007. Fully unsupervised discovery of concept-specific relationships by Web mining. *ACL '07*.
- Davidov, D., Rappoport, A., 2008. Classification of relationships between nominals using pattern clusters. *ACL '08*.
- Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., and Yates, A., 2004. Methods for domain-independent information extraction from the web: An experimental comparison. *AAAI 04*
- Gabrilovich, E., Markovitch, S., 2005. Feature generation for text categorization using world knowledge. *IJCAI 2005*.
- Girju, R., Giuglea, A., Olteanu, M., Fortu, O., Bolohan, O., and Moldovan, D., 2004. Support vector machines applied to the classification of semantic relations in nominalized noun phrases. *HLT/NAACL Workshop on Computational Lexical Semantics*.
- Girju, R., Moldovan, D., Tatu, M., and Antohe, D., 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19(4):479-496.
- Girju, R., Badulescu, A., and Moldovan, D., 2006. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1).
- Girju, R., Hearst, M., Nakov, P., Nastase, V., Szpakowicz, S., Turney, P., and Yuret, D., 2007. Task 04: Classification of semantic relations between nominal at SemEval 2007. *ACL '07 SemEval Workshop*.
- Hasegawa, T., Sekine, S., and Grishman, R., 2004. Discovering relations among named entities from large corpora. *ACL '04*.
- Hassan, H., Hassan, A. and Emam, O., 2006. Unsupervised information extraction approach using graph mutual reinforcement. *EMNLP '06*.
- Hearst, M., 1992. Automatic acquisition of hyponyms from large text corpora. *COLING '92*
- Lin, D., Pantel, P., 2002. Concept discovery from text. *COLING 02*.
- Moldovan, D., Badulescu, A., Tatu, M., Antohe, D., Girju, R., 2004. Models for the semantic classification of noun phrases. *HLT-NAACL '04 Workshop on Computational Lexical Semantics*.
- Nastase, V., Szpakowicz, S., 2003. Exploring noun modifier semantic relations. *IWCS-5*.
- Pantel, P., Pennacchiotti, M., 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. *COLING-ACL 2006*.
- Pantel, P., Ravichandran, D. and Hovy, E.H., 2004. Towards terascale knowledge acquisition. *COLING '04*.
- Pasca, M., Lin, D., Bigham, J., Lifchits A., Jain, A., 2006. Names and similarities on the web: fact extraction in the fast lane. *COLING-ACL '06*.
- Rosenfeld, B., Feldman, R., 2007. Clustering for unsupervised relation identification. *CIKM '07*.
- Snow, R., Jurafsky, D., Ng, A.Y., 2006. Semantic taxonomy induction from heterogeneous evidence. *COLING-ACL '06*.
- Strube, M., Ponzetto, S., 2006. WikiRelate! computing semantic relatedness using Wikipedia. *AAAI '06*.
- Suchanek, F., Ifrim, G., and Weikum, G., 2006. LEILA: learning to extract information by linguistic analysis. *COLING-ACL '06 Ontology Learning & Population Workshop*.
- Tatu, M., Moldovan, D., 2005. A semantic approach to recognizing textual entailment. *HLT/EMNLP 2005*.
- Turney, P., 2005. Measuring semantic similarity by latent relational analysis. *IJCAI '05*.
- Turney, P., Littman, M., 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning*(60):1-3:251-278.
- Turney, P., 2006. Expressing implicit semantic relations without supervision. *COLING-ACL '06*.
- Widdows, D., Dorow, B., 2002. A graph model for unsupervised lexical acquisition. *COLING '02*.



# Improving Search Results Quality by Customizing Summary Lengths

**Michael Kaisser**  
University of Edinburgh  
2 Buccleuch Place  
Edinburgh EH8 9LW  
*m.kaisser@sms.ed.ac.uk*

**Marti A. Hearst**  
UC Berkeley  
102 South Hall  
Berkeley, CA 94705  
*hearst@ischool.berkeley.edu*

**John B. Lowe**  
Powerset, Inc.  
475 Brannan St.  
San Francisco, CA 94107  
*johnblowe@gmail.com*

## Abstract

Web search engines today typically show results as a list of titles and short snippets that summarize how the retrieved documents are related to the query. However, recent research suggests that longer summaries can be preferable for certain types of queries. This paper presents empirical evidence that judges can predict appropriate search result summary lengths, and that perceptions of search result quality can be affected by varying these result lengths. These findings have important implications for search results presentation, especially for natural language queries.

## 1 Introduction

Search results listings on the web have become standardized as a list of information summarizing the retrieved documents. This summary information is often referred to as the document's *surrogate* (Marchionini et al., 2008).

In older search systems, such as those used in news and legal search, the document surrogate typically consisted of the title and important metadata, such as date, author, source, and length of the article, as well as the document's manually written abstract. In most cases, the full text content of the document was not available to the search engine and so no extracts could be made.

In web search, document surrogates typically show the web page's title, a URL, and information extracted from the full text contents of the document. This latter part is referred to by several different names, including *summary*, *abstract*, *extract*,

and *snippet*. Today it is standard for web search engines to show these summaries as one or two lines of text, often with ellipses separating sentence fragments. However, there is evidence that the ideal result length is often longer than the standard snippet length, and that furthermore, result length depends on the type of answer being sought.

In this paper, we systematically examine the question of search result length preference, comparing different result lengths for different query types. We find evidence that desired answer length is sensitive to query type, and that for some queries longer answers are judged to be of higher quality.

In the following sections we summarize the related work on result length variation and on query topic classification. We then describe two studies. In the first, judges examined queries and made predictions about the expected answer types and the ideal answer lengths. In the second study, judges rated answers of different lengths for these queries. The studies find evidence supporting the idea that different query types are best answered with summaries of different lengths.

## 2 Related Work

### 2.1 Query-biased Summaries

In the early days of the web, the result summary consisted of the first few lines of text, due both to concerns about intellectual property, and because often that was the only part of the full text that the search engines retained from their crawls. Eventually, search engines started showing what are known variously as *query-biased summaries*, *keyword-in-*

context (KWIC) extractions, and *user-directed summaries* (Tombros and Sanderson, 1998). In these summaries, sentence fragments, full sentences, or groups of sentences that contain query terms are extracted from the full text. Early versions of this idea were developed in the Snippet Search tool (Pedersen et al., 1991) and the Superbook tool's Table-of-Contents view (Egan et al., 1989).

A query-biased summary shows sentences that summarize the ways the query terms are used within the document. In addition to showing which subsets of query terms occur in a retrieved document, this display also exposes the context in which the query terms appear with respect to one another.

Research suggests that query-biased summaries are superior to showing the first few sentences from documents. Tombros & Sanderson (1998), in a study with 20 participants using TREC *ad hoc* data, found higher precision and recall and higher subjective preferences for query-biased summaries over summaries showing the first few sentences. Similar results for timing and subjective measurements were found by White et al. (2003) in a study with 24 participants. White et al. (2003) also describe experiments with different sentence selection mechanisms, including giving more weight to sentences that contained query words along with text formatting.

There are significant design questions surrounding how best to formulate and display query-biased summaries. As with standard document summarization and extraction, there is an inherent trade-off between showing long, informative summaries and minimizing the screen space required by each search hit. There is also a tension between showing short snippets that contain all or most of the query terms and showing coherent stretches of text. If the query terms do not co-occur near one another, then the extract has to become very long if full sentences and all query terms are to be shown. Many web search engine snippets compromise by showing fragments instead of sentences.

## 2.2 Studies Comparing Results Lengths

Recently, a few studies have analyzed the results of varying search summary length.

In the question-answering context (as opposed to general web search), Lin et al. (2003) conducted

a usability study with 32 computer science students comparing four types of answer context: exact answer, answer-in-sentence, answer-in-paragraph, and answer-in-document. To remove effects of incorrect answers, they used a system that produced only correct answers, drawn from an online encyclopedia. Participants viewed answers for 8 question scenarios. Lin et al. (2003) found no significant differences in task completion times, but they did find differences in subjective responses. Most participants (53%) preferred paragraph-sized chunks, noting that a sentence wasn't much more information beyond the exact answer, and a full document was often-times too long. That said, 23% preferred full documents, 20% preferred sentences, and one participant preferred exact answer, thus suggesting that there is considerable individual variation.

Paek et al. (2004) experimented with showing differing amounts of summary information in results listings, controlling the study design so that only one result in each list of 10 was relevant. For half the test questions, the target information was visible in the original snippet, and for the other half, the participant needed to use their mouse to view more information from the relevant search result. They compared three interface conditions:

- (i) a standard search results listing, in which a mouse click on the title brings up the full text of the web page,
- (ii) "instant" view, for which a mouseclick expanded the document summary to show additional sentences from the document, and those sentences contained query terms and the answer to the search task, and
- (iii) a "dynamic" view that responded to a mouse hover, and dynamically expanded the summary with a few words at a time.

Eleven out of 18 participants preferred instant view over the other two views, and on average all participants produced faster and more accurate results with this view. Seven participants preferred dynamic view over the others, but many others found this view disruptive. The dynamic view suffered from the problem that, as the text expanded, the mouse no longer covered the selected results, and

so an unintended, different search result sometimes started to expand. Notably, none of the participants preferred the standard results listing view.

Cutrell & Guan (2007), compared search summaries of varying length: short (1 line of text), medium (2-3 lines) and long (6-7 lines) using search engine-produced snippets (it is unclear if the summary text was contiguous or included ellipses). They also compared 6 navigational queries (where the goal is to find a website's homepage), with 6 informational queries (e.g., "find when the Titanic set sail for its only voyage and what port it left from," "find out how long the Las Vegas monorail is"). In a study with 22 participants, they found that participants were 24 seconds faster on average with the long view than with the short and medium view. They also found that participants were 10 seconds slower on average with the long view for the navigational tasks. They present eye tracking evidence which suggests that on the navigational task, the extra text distracts the eye from the URL. They did not report on subjective responses to the different answer lengths.

Rose et al. (2007) varied search results summaries along several dimensions, finding that text chopiness and sentence truncation had negative effects, and genre cues had positive effects. They did not find effects for varying summary length, but they only compared relatively similar summary lengths (2 vs. 3 vs. 4 lines long).

### 2.3 Categorizing Questions by Expected Answer Types

In the field of automated question-answering, much effort has been expended on automatically determining the kind of answer that is expected for a given question. The candidate answer types are often drawn from the types of questions that have appeared in the TREC Question Answering track (Voorhees, 2003). For example, the Webclopedia project created a taxonomy of 180 types of question targets (Hovy et al., 2002), and the FALCON project (Harabagiu et al., 2003) developed an answer taxonomy with 33 top level categories (such as PERSON, TIME, REASON, PRODUCT, LOCATION, NUMERICAL VALUE, QUOTATION), and these were further refined into an unspecified number of additional categories. Ramakrishnan et al.

(2004) show an automated method for determining expected answer types using syntactic information and mapping query terms to WordNet.

### 2.4 Categorizing Web Queries

A different line of research is the query log categorization problem. In query logs, the queries are often much more terse and ill-defined than in the TREC QA track, and, accordingly, the taxonomies used to classify what is called the query intent have been much more general.

In an attempt to demonstrate how information needs for web search differ from the assumptions of pre-web information retrieval systems, Broder (2002) created a taxonomy of web search goals, and then estimated frequency of such goals by a combination of an online survey (3,200 responses, 10% response rate) and a manual analysis of 1,000 query from the AltaVista query logs. This taxonomy has been heavily influential in discussions of query types on the Web.

Rose & Levinson (2004) followed up on Broder's work, again using web query logs, but developing a taxonomy that differed somewhat from Broder's. They manually classified a set of 1,500 AltaVista search engine log queries. For two sets of 500 queries, the labeler saw just the query and the retrieved documents; for the third set the labeler also saw information about which item(s) the searcher clicked on. They found that the classifications that used the extra information about clickthrough did not change the proportions of assignments to each category. Because they did not directly compare judgments with and without click information on the same queries, this is only weak evidence that query plus retrieved documents is sufficient to classify query intent.

Alternatively, queries from web query logs can be classified according to the *topic* of the query, independent of the type of information need. For example, a search involving the topic of weather can consist of the simple information need of looking at today's forecast, or the rich and complex information need of studying meteorology. Over many years, Spink & Jansen et al. (2006; 2007) have manually analyzed samples of query logs to track a number of different trends. One of the most notable is the change in topic mix. As an alternative to man-

ual classification of query topics, Shen et al. (2005) described an algorithm for automatically classifying web queries into a set of pre-defined topics. More recently, Broder et al. (2007) presented a highly accurate method (around .7 F-score) for classifying short, rare queries into a taxonomy of 6,000 categories.

### 3 Study Goals

Related work suggests that longer results are preferable, but not for all query types. The goal of our efforts was to determine preferred result length for search results, depending on type of query. To do this, we performed two studies:

1. We asked a set of judges to categorize a large set of web queries according to their expected preferred response type and expected preferred response length.
2. We then developed high-quality answer passages of different lengths for a subset of these queries by selecting appropriate passages from the online encyclopedia Wikipedia, and asked judges to rate the quality of these answers.

The results of this study should inform search interface designers about what the best presentation format is.

#### 3.1 Using Mechanical Turk

For these studies, we make use of a web service offered by Amazon.com called Mechanical Turk, in which participants (called “turkers”) are paid small sums of money in exchange for work on “Human Intelligence tasks” (HITs).<sup>1</sup> These HITs are generated from an XML description of the task created by the investigator (called a “requester”). The participants can come from any walk of life, and their identity is not known to the requesters. We have in past work found the results produced by these judges to be of high quality, and have put into place various checks to detect fraudulent behavior. Other researchers have investigated the efficacy of language

<sup>1</sup>Website: <http://www.mturk.com>. For experiment 1, approximately 38,000 HITs were completed at a cost of about \$1,500. For experiment 2, approximately 7,300 HITs were completed for about \$170. Turkers were paid between \$.01 and \$.05 per HIT depending on task complexity; Amazon imposes additional charges.

- 
1. Person(s)
  2. Organization(s)
  3. Time(s) (date, year, time span etc.)
  4. Number or Quantity
  5. Geographic Location(s) (e.g., city, lake, address)
  6. Place(s) (e.g., “the White House”, “at a supermarket”)
  7. Obtain resource online (e.g., movies, lyrics, books, magazines, knitting patterns)
  8. Website or URL
  9. Purchase and product information
  10. Gossip and celebrity information
  11. Language-related (e.g., translations, definitions, crossword puzzle answers)
  12. General information about a topic
  13. Advice
  14. Reason or Cause, Explanation
  15. Yes/No, with or without explanation or evidence
  16. Other
  17. Unjudgable
- 

Table 1: Allowable responses to the question: “What sort of result or results does the query ask for?” in the first experiment.

- 
1. A word or short phrase
  2. A sentence
  3. One or more paragraphs (i.e. at least several sentences)
  4. An article or full document
  5. A list
  6. Other, or some combination of the above
- 

Table 2: Allowable responses to the question: “How long is the best result for this query?” in the first experiment.

annotation using this service and have found that the results are of high quality (Su et al., 2007).

#### 3.2 Estimating Ideal Answer Length and Type

We developed a set of 12,790 queries, drawn from Powerset’s in house query database which contains representative subsets of queries from different search engines’ query logs, as well as hand-edited query sets used for regression testing. There are a disproportionately large number of natural language queries in this set compared with query sets from typical keyword engines. Such queries are often complete questions and are sometimes grammatical fragments (e.g., “date of next US election”) and so are likely to be amenable to interesting natural language processing algorithms, which is an area of in-

Answer Type	Answer Length					
	Phrase	Sentence	Paragraphs	Article	List	Combination
Person	1,362	735	570	378	419	68
Organization	153	172	295	165	432	51
Time	964	486	176	65	126	21
Number	2,075	964	362	88	158	50
GeoLocation	552	399	269	126	389	78
Place	128	121	173	87	295	33
Resource	104	136	733	273	959	256
Website	243	168	101	52	297	61
Purchase	200	318	780	295	1,231	276
Gossip	86	133	366	156	85	51
NatLang	946	479	186	21	171	26
GeneralInfo	396	861	3,197	3,244	1,075	359
Advice	43	164	1,257	1,086	357	151
ReasonCause	50	102	755	546	88	69
YesNo	392	281	306	73	12	8
Other	115	61	140	157	88	29
Unjudgable	59	47	36	18	18	556

Figure 1: Results of the first experiment. The y-axis shows the semantic type of the predicted answer, in the same order as listed in Table 1; the x-axis shows the preferred length as listed in Table 2. Three bars with length greater than 1,500 are trimmed to the maximum size to improve readability (GeneralInfo/Paragraphs, GeneralInfo/Article, and Number/Phrase).

terest of our research. The average number of words per query (as determined by white space separation) was 5.8 (sd. 2.9) and the average number of characters (including punctuation and white space) was 32.3 (14.9). This is substantially longer than the current average for web search query, which was approximately 2.8 in 2005 (Jansen et al., 2007); this is due to the existence of natural language queries.

Judges were asked to classify each query according to its expected response type into one of 17 categories (see Table 1). These categories include answer types used in question answering research as well as (to better capture the diverse nature of web queries) several more general response types such as *Advice* and *General Information*. Additionally, we asked judges to anticipate what the best result length would be for the query, as shown in Table 2.

Each of the 12,790 queries received three assessments by MTurk judges. For answer types, the number of times all three judges agreed was 4537 (35.4%); two agreed 6030 times (47.1%), and none

agreed 2223 times (17.4%). Not surprisingly, there was significant overlap between the label *General-Info* and the other categories. For answer length estimations, all three judges agreed in 2361 cases (18.5%), two agreed in 7210 cases (56.4%) and none 3219 times (25.2%).

Figure 1 summarizes expected length judgments by estimated answer category. Distribution of the length categories differs a great deal across the individual expected response categories. In general, the results are intuitive: judges preferred short responses for “precise queries” (e.g., those asking for numbers) and they preferred longer responses for queries in broad categories like *Advice* or *General-Info*. But some results are less intuitive: for example, judges preferred different response lengths for queries categorized as *Person* and *Organization* – in fact for the latter the largest single selection made was *List*. Reviewing the queries for these two categories, we note that most queries about organizations in our collection asked for companies

length type	average	std dev
Word or Phrase	38.1	25.8
Sentence	148.1	71.4
Paragraph	490.5	303.1
Section	1574.2	1251.1

Table 3: Average number of characters for each answer length type for the stimuli used in the second experiment.

(e.g. “around the world travel agency”) and for these there usually is more than one correct answer, whereas the queries about persons (“CEO of microsoft”) typically only had one relevant answer. The results of this table show that there are some trends but not definitive relationships between query type (as classified in this study) and expected answer length. More detailed classifications might help resolve some of the conflicts.

### 3.3 Result Length Study

The purpose of the second study was twofold: first, to see if doing a larger study confirms what is hinted at in the literature: that search result lengths longer than the standard snippet may be desirable for at least a subset of queries. Second, we wanted to see if judges’ predictions of desirable results lengths would be confirmed by other judges’ responses to search results of different lengths.

#### 3.3.1 Method

It has been found that obtaining judges’ agreement on intent of a query from a log can be difficult (Rose and Levinson, 2004; Kellar et al., 2007). In order to make the task of judging query relevance easier, for the next phase of the study we focused on only those queries for which all three assessors in the first experiment agreed both on the category label and on the estimated ideal length. There were 1099 such high-confidence queries, whose average number of words was 6.3 (2.9) and average number of characters was 34.5 (14.3).

We randomly selected a subset of the high-agreement queries from the first experiment and manually excluded queries for which it seemed obvious that no responses could be found in Wikipedia. These included queries about song lyrics, since intellectual property restrictions prevent these being posted, and crossword puzzle questions such as “a four letter word for water.”

The remaining set contained 170 queries. MTurk annotators were asked to find one good text passage (in English) for each query from the Wikipedia online encyclopedia. They were also asked to subdivide the text of this answer into each of the following lengths: a word or phrase, a sentence, a paragraph, a section or an entire article.<sup>2</sup> Thus, the shorter answer passages are subsumed by the longer ones.

Table 3 shows the average lengths and standard deviations of each result length type. Table 4 contains sample answers for the shorter length formats for one query. For 24 of the 170 queries the annotators could not find a suitable response in Wikipedia, e.g., “How many miles between NY and Milwaukee?” We collected two to five results for each of the remaining 146 queries and manually chose the best of these answer passages. Note that, by design, all responses were factually correct; they only differed in their length.

Ten MTurk judges saw each query/answer length pair, and for each of these, were told: “Below you see a search engine query and a possible response. We would like you to give us your opinion about the response. We are especially interested in the length of the response. Is it suitable for the query? Is there too much or not enough information? Please rate the response on a scale from 0 (very bad response) to 10 (very good response).” There were 124 judges in total; of these, 16 did more than 146 HITS, meaning they saw the same query more than one time (but with different lengths). Upon examination of the results, we determined that two of these high-volume judges were not trying to do the task properly, and so we dropped their judgments from the final analysis.

#### 3.3.2 Results

Our results show that judges prefer results of different lengths, depending on the query. The results also suggest that judges’ estimates of a preferred result length in the first experiment are accurate predictors when there is strong agreement among them. Figure 2 shows in four diagrams

<sup>2</sup>Note the slight difference between the length categories in the first and second experiment: The *List* and *Other* options were dropped for the second experiment because we wanted to concentrate on textual length. Additionally, to provide more than one option between *Sentence* and *Article*, the category *One or more paragraphs* was split up into two: *(One) Paragraph* and *(One) Section*.

<b>query</b>	Who was the first person to scale K2?
<b>Paragraph</b>	An Italian expedition finally succeeded in ascending to the summit of K2 on July 31, 1954. The expedition was led by Ardito Desio, although the two climbers who actually reached the top were Lino Lacedelli and Achille Compagnoni. The team included a Pakistani member, Colonel Muhammad Ata-ullah. He had been a part of an earlier 1953 American expedition which failed to make the summit because of a storm which killed a key climber, Art Gilkey. On the expedition also was the famous Italian climber Walter Bonatti. He proved vital to the expeditions success in that he carried vital oxygen to 26,600ft for Lacedelli and Compagnoni. His dramatic bivouac, at that altitude with the equipment, wrote another chapter in the saga of Himalayan climbing.
<b>Sentence</b>	The expedition was led by Ardito Desio, although the two climbers who actually reached the top were Lino Lacedelli and Achille Compagnoni.
<b>Phrase</b>	Lino Lacedelli and Achille Compagnoni

Table 4: Sample answers of differing lengths used as input for the second study. Note that the shorter answers are contained in the longer ones. For the full article case, judges were asked to follow a hyperlink to an article.

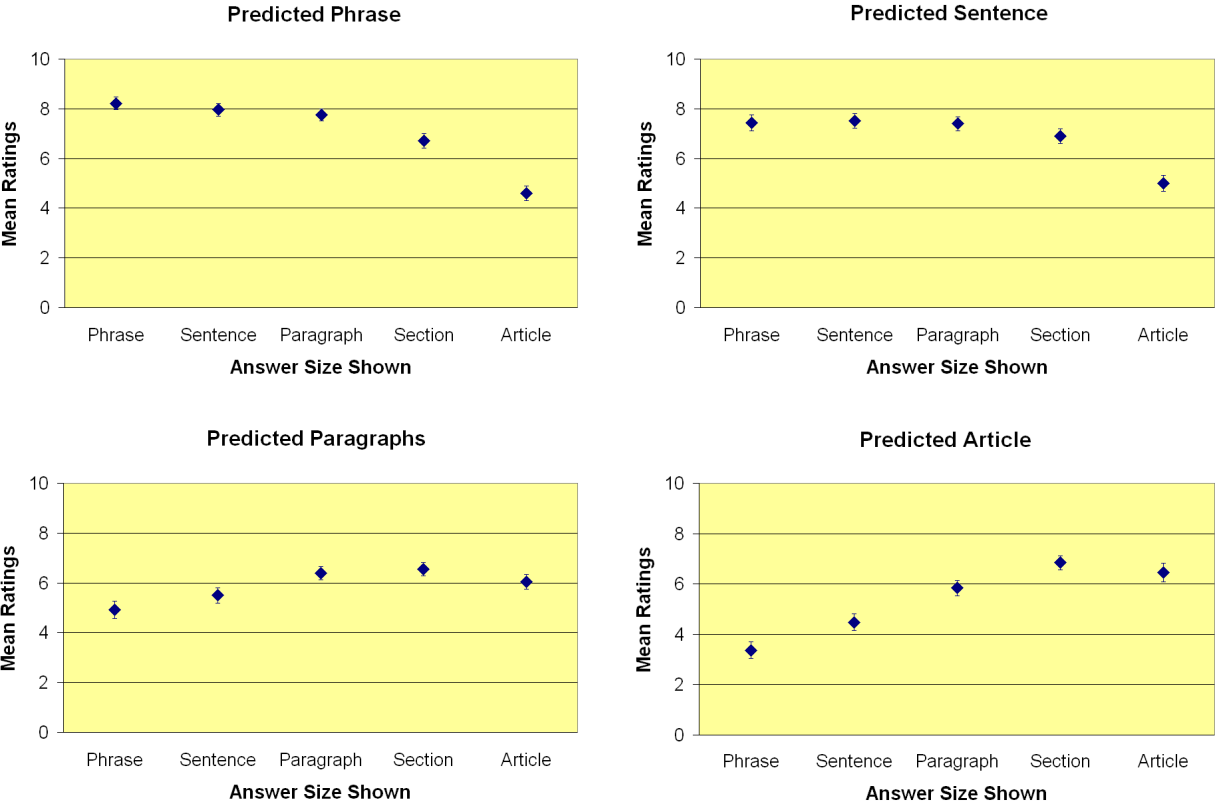


Figure 2: Results of the second experiment, where each query/answer-length pair was assessed by 8–10 judges using a scale of 0 (‘very bad’) to 10 (‘very good’). Marks indicate means and standard errors. The top left graph shows responses of different lengths for queries that were classified as *best answered with a phrase* in the first experiment. The upper right shows responses for queries predicted to be *best answered with a sentence*, lower left for *best answered with one or more paragraphs* and lower right for *best answered with an article*.

	Slope	Std. Error	p-value
Phrase	-0.850	0.044	< 0.0001
Sentence	-0.550	0.050	< 0.0001
Paragraph	0.328	0.049	< 0.0001
Article	0.856	0.053	< 0.0001

Table 5: Results of unweighted linear regression on the data for the second experiment, which was separated into four groups based on the predicted preferred length.

how queries assigned by judges to one of the four length categories from the first experiment were judged when presented with responses of the five answer lengths from the second experiment. The graphs show the means and standard error of the judges' scores across all queries for each predicted-length/presented-length combination.

In order to test whether these results are significant we performed four separate linear regressions; one for each of the predicted preferred length categories. The snippet length, the independent variable, was coded as 1-5, shortest to longest. The score for each query-snippet pair is the dependent variable. Table 5 shows that for each group there is evidence to reject the null hypothesis that the slope is equal to 0 at the 99% confidence level. High scores are associated with shorter snippet lengths for queries with predicted preferred length *phrase* or *sentence* and also with longer snippet lengths for queries with predicted preferred length *paragraphs* or *article*. These associations are strongest for the queries with the most extreme predicted preferred lengths (*phrase* and *article*).

Our results also suggest the intuition that the best answer lengths do not form strictly distinct classes, but rather lie on a continuum. If the ideal response is from a certain category (e.g., a sentence), returning a result from an adjacent category (a phrase or a paragraph) is not strongly penalized by judges, whereas retuning a result from a category further up or down the scale (an article) is.

One potential drawback of this study format is that we do not show judges a list of results for queries, as is standard in search engines, and so they do not experience the tradeoff effect of longer results requiring more scrolling if the desired answer is not shown first. However, the earlier results of Cutrell & Guan (2007) and Paek et al. (2004) suggest that the

preference for longer results occurs even in contexts that require looking through multiple results. Another potential drawback of the study is that judges only view one relevant result; the effects of showing a list of long non-relevant results may be more negative than that of showing short non-relevant results; this study would not capture that effect.

## 4 Conclusions and Future Work

Our studies suggest that different queries are best served with different response lengths (Experiment 1), and that for a subset of especially clear queries, human judges can predict the preferred result lengths (Experiment 2). The results furthermore support the contention that standard results listings are too short in many cases, at least assuming that the summary shows information that is relevant for the query. These findings have important implications for the design of search results presentations, suggesting that as user queries become more expressive, search engine results should become more responsive to the type of answer desired. This may mean showing more context in the results listing, or perhaps using more dynamic tools such as expand-on-mouseover to help answer the query in place.

The obvious next step is to determine how to automatically classify queries according to their predicted result length and type. For classifying according to expected length, we have run some initial experiments based on unigram word counts which correctly classified 78% of 286 test queries (on 805 training queries) into one of three length bins. We plan to pursue this further in future work. For classifying according to type, as discussed above, most automated query classification for web logs have been based on the topic of the query rather than on the intended result type, but the question answering literature has intensively investigated how to predict appropriate answer types. It is likely that the techniques from these two fields can be productively combined to address this challenge.

**Acknowledgments.** This work was supported in part by Powerset, Inc., and in part by Microsoft Research through the MSR European PhD Scholarship Programme. We would like to thank Susan Gruber and Bonnie Webber for their helpful comments and suggestions.



## References

- A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. 2007. Robust classification of rare queries using web knowledge. *Proceedings of SIGIR 2007*.
- A. Broder. 2002. A taxonomy of web search. *ACM SIGIR Forum*, 36(2):3–10.
- E. Cutrell and Z. Guan. 2007. What Are You Looking For? An Eye-tracking Study of Information Usage in Web Search. *Proceedings of ACM SIGCHI 2007*.
- D.E. Egan, J.R. Remde, L.M. Gomez, T.K. Landauer, J. Eberhardt, and C.C. Lochbaum. 1989. Formative design evaluation of Superbook. *ACM Transactions on Information Systems (TOIS)*, 7(1):30–57.
- S.M. Harabagiu, S.J. Maiorano, and M.A. Pasca. 2003. Open-domain textual question answering techniques. *Natural Language Engineering*, 9(03):231–267.
- E. Hovy, U. Hermjakob, and D. Ravichandran. 2002. A question/answer typology with surface text patterns. *Proceedings of the second international conference on Human Language Technology Research*, pages 247–251.
- B.J. Jansen and Spink. 2006. How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing and Management*, 42(1):248–263.
- B.J. Jansen, A. Spink, and S. Koshman. 2007. Web searcher interaction with the Dogpile.com metasearch engine. *Journal of the American Society for Information Science and Technology*, 58(5):744–755.
- M. Kellar, C. Watters, and M. Shepherd. 2007. A Goal-based Classification of Web Information Tasks. *JASIST*, 43(1).
- J. Lin, D. Quan, V. Sinha, K. Bakshi, D. Huynh, B. Katz, and D.R. Karger. 2003. What Makes a Good Answer? The Role of Context in Question Answering. *Human-Computer Interaction (INTERACT 2003)*.
- G. Marchionini, R.W. White, and Marchionini. 2008. Find What You Need, Understand What You Find. *Journal of Human-Computer Interaction (to appear)*.
- T. Paek, S.T. Dumais, and R. Logan. 2004. WaveLens: A new view onto internet search results. *Proceedings on the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 727–734.
- J. Pedersen, D. Cutting, and J. Tukey. 1991. Snippet search: A single phrase approach to text access. *Proceedings of the 1991 Joint Statistical Meetings*.
- G. Ramakrishnan and D. Paranjpe. 2004. Is question answering an acquired skill? *Proceedings of the 13th international conference on World Wide Web*, pages 111–120.
- D.E. Rose and D. Levinson. 2004. Understanding user goals in web search. *Proceedings of the 13th international conference on World Wide Web*, pages 13–19.
- D.E. Rose, D. Orr, and R.G.P. Kantamneni. 2007. Summary attributes and perceived search quality. *Proceedings of the 16th international conference on World Wide Web*, pages 1201–1202.
- D. Shen, R. Pan, J.T. Sun, J.J. Pan, K. Wu, J. Yin, and Q. Yang. 2005. Q2C@UST: our winning solution to query classification in KDDCUP 2005. *ACM SIGKDD Explorations Newsletter*, 7(2):100–110.
- Q. Su, D. Pavlov, J. Chow, and W. Baker. 2007. Internet-Scale Collection of Human-Reviewed Data. *Proceedings of WWW 2007*.
- A. Tombros and M. Sanderson. 1998. Advantages of query biased summaries in information retrieval. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 2–10.
- E.M. Voorhees. 2003. Overview of the TREC 2003 Question Answering Track. *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*.
- R.W. White, J. Jose, and I. Ruthven. 2003. A task-oriented study on the influencing effects of query-biased summarisation in web searching. *Information Processing and Management*, 39(5):707–733.

# Using Conditional Random Fields to Extract Contexts and Answers of Questions from Online Forums

Shilin Ding<sup>†\*</sup> Gao Cong<sup>§†</sup> Chin-Yew Lin<sup>‡</sup> Xiaoyan Zhu<sup>†</sup>

<sup>†</sup>Department of Computer Science and Technology, Tsinghua University, Beijing, China

<sup>§</sup>Department of Computer Science, Aalborg University, Denmark

<sup>‡</sup>Microsoft Research Asia, Beijing, China

dingsl@gmail.com gaocong@cs.aau.dk

cyl@microsoft.com zxy-dcs@tsinghua.edu.cn

## Abstract

Online forum discussions often contain vast amounts of questions that are the focuses of discussions. Extracting contexts and answers together with the questions will yield not only a coherent forum summary but also a valuable QA knowledge base. In this paper, we propose a general framework based on Conditional Random Fields (CRFs) to detect the contexts and answers of questions from forum threads. We improve the basic framework by Skip-chain CRFs and 2D CRFs to better accommodate the features of forums for better performance. Experimental results show that our techniques are very promising.

## 1 Introduction

Forums are web virtual spaces where people can ask questions, answer questions and participate in discussions. The availability of vast amounts of thread discussions in forums has promoted increasing interests in knowledge acquisition and summarization for forum threads. Forum thread usually consists of an initiating post and a number of reply posts. The initiating post usually contains several questions and the reply posts usually contain answers to the questions and perhaps new questions. Forum participants are not physically co-present, and thus reply may not happen immediately after questions are posted. The asynchronous nature and multi-participants make multiple questions and answers

\*This work was done when Shilin Ding was a visiting student at the Microsoft Research Asia

<sup>†</sup>This work was done when Gao Cong worked as a researcher at the Microsoft Research Asia.

<context id=1>**S1:** *Hi I am looking for a pet friendly hotel in Hong Kong because all of my family is going there for vacation.* **S2:** *my family has 2 sons and a dog.*</context> <question id=1>**S3:** *Is there any recommended hotel near Sheung Wan or Tsing Sha Tsui?*</question> <context id=2,3>**S4:** *We also plan to go shopping in Causeway Bay.*</context> <question id=2>**S5:** *What's the traffic situation around those commercial areas?*</question> <question id=3>**S6:** *Is it necessary to take a taxi?*</question>. **S7:** *Any information would be appreciated.*  
<answer qid=1>**S8:** *The Comfort Lodge near Kowloon Park allows pet as I know, and usually fits well within normal budget.* **S9:** *It is also conveniently located, nearby the Kowloon railway station and subway.*</answer>  
<answer qid=2,3> **S10:** *It's very crowd in those areas, so I recommend MTR in Causeway Bay because it is cheap to take you around* </answer>

Figure 1: An example thread with question-context-answer annotated

interweaved together, which makes it more difficult to summarize.

In this paper, we address the problem of detecting the contexts and answers from forum threads for the questions identified in the same threads. Figure 1 gives an example of a forum thread with questions, contexts and answers annotated. It contains three *question* sentences, S3, S5 and S6. Sentences S1 and S2 are *contexts* of *question* 1 (S3). Sentence S4 is the *context* of *questions* 2 and 3, but not 1. Sentence S8 is the answer to question 3. (S4-S5-S10) is one example of question-context-answer triple that we want to detect in the thread. As shown in the example, a forum question usually requires contextual information to provide background or constraints.

Moreover, it sometimes **needs contextual information to provide explicit link to its answers**. For example, S8 is an answer of *question 1*, but they cannot be linked with any common word. Instead, S8 shares word *pet* with S1, which is a context of *question 1*, and thus S8 could be linked with *question 1* through S1. We call contextual information the *context* of a question in this paper.

A summary of forum threads in the form of question-context-answer can not only highlight the main content, but also provide a user-friendly organization of threads, which will make the access to forum information easier.

Another motivation of detecting contexts and answers of the questions in forum threads is that it could be used to enrich the knowledge base of community-based question and answering (CQA) services such as Live QnA and Yahoo! Answers, where *context* is comparable with the question description while *question* corresponds to the question title. For example, there were about 700,000 questions in the Yahoo! Answers travel category as of January 2008. We extracted about 3,000,000 travel related questions from six online travel forums. One would expect that a CQA service with large QA data will attract more users to the service. To enrich the knowledge base, not only the answers, but also the contexts are critical; otherwise the answer to a question such as *How much is the taxi* would be useless without context in the database.

However, it is challenging to detecting contexts and answers for questions in forum threads. We assume the questions have been identified in a forum thread using the approach in (Cong et al., 2008). Although identifying questions in a forum thread is also nontrivial, it is beyond the focus of this paper.

First, detecting contexts of a question is important and non-trivial. We found that 74% of questions in our corpus, which contain 1,064 questions from 579 forum threads about travel, need contexts. However, relative position information is far from adequate to solve the problem. For example, in our corpus 63% of sentences preceding questions are contexts and they only represent 34% of all correct contexts. To effectively detect contexts, the dependency between sentences is important. For example in Figure 1, both S1 and S2 are contexts of *question 1*. S1 could be labeled as context based on word similarity, but it

is not easy to link S2 with the question directly. S1 and S2 are linked by the common word *family*, and thus S2 can be linked with *question 1* through S1. The challenge here is how to model and utilize the dependency for context detection.

Second, it is difficult to link answers with questions. In forums, multiple questions and answers can be discussed in parallel and are interweaved together while the reply relationship between posts is usually unavailable. To detect answers, we need to handle two kinds of dependencies. One is the dependency relationship between contexts and answers, which should be leveraged especially when questions alone do not provide sufficient information to find answers; the other is the dependency between answer candidates (similar to sentence dependency described above). The challenge is how to model and utilize these two kinds of dependencies.

In this paper we propose a novel approach for detecting contexts and answers of the questions in forum threads. To our knowledge this is the first work on this. We make the following contributions:

First, we employ Linear Conditional Random Fields (CRFs) to identify contexts and answers, which can capture the relationships between contiguous sentences.

Second, we also found that context is very important for answer detection. To capture the dependency between contexts and answers, we introduce Skip-chain CRF model for answer detection. We also extend the basic model to 2D CRFs to model dependency between contiguous questions in a forum thread for context and answer identification.

Finally, we conducted experiments on forum data. Experimental results show that 1) Linear CRFs outperform SVM and decision tree in both context and answer detection; 2) Skip-chain CRFs outperform Linear CRFs for answer finding, which demonstrates that **context improves answer finding**; 3) 2D CRF model improves the performance of Linear CRFs and the combination of 2D CRFs and Skip-chain CRFs achieves better performance for context detection.

The rest of this paper is organized as follows: The next section discusses related work. Section 3 presents the proposed techniques. We evaluate our techniques in Section 4. Section 5 concludes this paper and discusses future work.

## 2 Related Work

There is some research on summarizing discussion threads and emails. Zhou and Hovy (2005) segmented internet relay chat, clustered segments into subtopics, and identified responding segments of the first segment in each sub-topic by assuming the first segment to be focus. In (Nenkova and Bagga, 2003; Wan and McKeown, 2004; Rambow et al., 2004), email summaries were organized by extracting overview sentences as discussion issues. Carenini et al (2007) leveraged both quotation relation and clue words for email summarization. In contrast, given a forum thread, we extract questions, their contexts, and their answers as summaries.

Shrestha and McKeown (2004)’s work on email summarization is closer to our work. They used RIPPER as a classifier to detect interrogative questions and their answers and used the resulting question and answer pairs as summaries. However, it did not consider contexts of questions and dependency between answer sentences.

We also note the existing work on extracting knowledge from discussion threads. Huang et al.(2007) used SVM to extract *input-reply* pairs from forums for chatbot knowledge. Feng et al. (2006a) used cosine similarity to match students’ query with reply posts for discussion-bot. Feng et al. (2006b) identified the most important message in online classroom discussion board. Our problem is quite different from the above work.

Detecting context for question in forums is related to the context detection problem raised in the QA roadmap paper commissioned by ARDA (Burger et al., 2006). To our knowledge, none of the previous work addresses the problem of context detection. The method of finding follow-up questions (Yang et al., 2006) from TREC context track could be adapted for context detection. However, the follow-up relationship is limited between questions while context is not. In our other work (Cong et al., 2008), we proposed a supervised approach for question detection and an unsupervised approach for answer detection without considering context detection.

Extensive research has been done in question-answering, e.g. (Berger et al., 2000; Jeon et al., 2005; Cui et al., 2005; Harabagiu and Hickl, 2006; Dang et al., 2007). They mainly focus on con-

structing answer for certain types of question from a large document collection, and usually apply sophisticated linguistic analysis to both questions and the documents in the collection. Soricut and Brill (2006) used statistical translation model to find the appropriate answers from their QA pair collections from FAQ pages for the posted question. In our scenario, we not only need to find answers for various types of questions in forum threads but also their contexts.

## 3 Context and Answer Detection

A question is a linguistic expression used by a questioner to request information in the form of an answer. The sentence containing request focus is called *question*. *Context* are the sentences containing constraints or background information to the question, while *answer* are that provide solutions. In this paper, we use sentences as the detection segment though it is applicable to other kinds of segments.

Given a thread and a set of  $m$  detected questions  $\{Q_i\}_{i=1}^m$ , our task is to find the contexts and answers for each question. We first discuss using Linear CRFs for context and answer detection, and then extend the basic framework to Skip-chain CRFs and 2D CRFs to better model our problem. Finally, we will briefly introduce CRF models and the features that we used for CRF model.

### 3.1 Using Linear CRFs

For ease of presentation, we focus on detecting contexts using Linear CRFs. The model could be easily extended to answer detection.

**Context detection.** As discussed in Introduction that context detection cannot be trivially solved by position information (See Section 4.2 for details), and dependency between sentences is important for context detection. Recall that in Figure 1, S2 could be labeled as context of Q1 if we consider the dependency between S2 and S1, and that between S1 and Q1, while it is difficult to establish connection between S2 and Q1 without S1. Table 1 shows that the correlation between the labels of contiguous sentences is significant. In other words, when a sentence  $Y_t$ ’s previous  $Y_{t-1}$  is not a context ( $Y_{t-1} \neq C$ ) then it is very likely that  $Y_t$  (i.e.  $Y_t \neq C$ ) is also not a context. It is clear that the candidate contexts are not independent and there are strong dependency rela-

Contiguous sentences	$y_t = C$	$y_t \neq C$
$y_{t-1} = C$	901	1,081
$y_{t-1} \neq C$	1,081	47,190

Table 1: Contingency table( $\chi^2 = 9,386, p\text{-value} < 0.001$ )

tionships between contiguous sentences in a thread. Therefore, a desirable model should be able to capture the dependency.

The context detection can be modeled as a classification problem. Traditional classification tools, e.g. SVM, can be employed, where each pair of question and candidate context will be treated as an instance. However, they cannot capture the dependency relationship between sentences.

To this end, we proposed a general framework to detect contexts and answers based on Conditional Random Fields (Lafferty et al., 2001) (CRFs) which are able to model the sequential dependencies between contiguous nodes. A CRF is an undirected graphical model  $G$  of the conditional distribution  $P(\mathbf{Y}|\mathbf{X})$ .  $\mathbf{Y}$  are the random variables over the labels of the nodes that are globally conditioned on  $\mathbf{X}$ , which are the random variables of the observations. (See Section 3.4 for more about CRFs)

Linear CRF model has been successfully applied in NLP and text mining tasks (McCallum and Li, 2003; Sha and Pereira, 2003). However, our problem cannot be modeled with Linear CRFs in the same way as other NLP tasks, where one node has a unique label. In our problem, each node (sentence) might have multiple labels since one sentence could be the context of multiple questions in a thread. Thus, it is difficult to find a solution to tag context sentences for all questions in a thread in single pass.

Here we assume that questions in a given thread are independent and are found, and then we can label a thread with  $m$  questions one-by-one in  $m$ -passes. In each pass, one question  $Q_i$  is selected as focus and each other sentence in the thread will be labeled as *context*  $C$  of  $Q_i$  or not using Linear CRF model. The graphical representations of Linear CRFs is shown in Figure2(a). The linear-chain edges can capture the dependency between two contiguous nodes. The observation sequence  $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t \rangle$ , where  $t$  is the number of sentences in a thread, represents predictors (to be described in Section 3.5), and the tag sequence  $\mathbf{y} = \langle y_1, \dots, y_t \rangle$ , where

$y_i \in \{C, P\}$ , determines whether a sentence is plain text  $P$  or context  $C$  of question  $Q_i$ .

**Answer detection.** Answers usually appear in the posts after the post containing the question. There are also strong dependencies between contiguous answer segments. Thus, position and similarity information alone are not adequate here. To cope with the dependency between contiguous answer segments, Linear CRFs model are employed as in context detection.

### 3.2 Leveraging Context for Answer Detection Using Skip-chain CRFs

We observed in our corpus 74% questions lack constraints or background information which are very useful to link question and answers as discussed in Introduction. Therefore, contexts should be leveraged to detect answers. The Linear CRF model can capture the dependency between contiguous sentences. However, it cannot capture the long distance dependency between contexts and answers.

One straightforward method of leveraging context is to detect contexts and answers in two phases, i.e. to first identify contexts, and then label answers using both the context and question information (e.g. the similarity between context and answer can be used as features in CRFs). The two-phase procedure, however, still cannot capture the non-local dependency between contexts and answers in a thread.

To model the long distance dependency between contexts and answers, we will use Skip-chain CRF model to detect context and answer together. Skip-chain CRF model is applied for entity extraction and meeting summarization (Sutton and McCallum, 2006; Galley, 2006). The graphical representation of a Skip-chain CRF given in Figure2(b) consists of two types of edges: linear-chain ( $y_{t-1}$  to  $y_t$ ) and skip-chain edges ( $y_i$  to  $y_j$ ).

Ideally, the skip-chain edges will establish the connection between candidate pairs with high probability of being context and answer of a question. To introduce skip-chain edges between any pairs of non-contiguous sentences will be computationally expensive, and also introduce noise. To make the cardinality and number of cliques in the graph manageable and also eliminate noisy edges, we would like to generate edges only for sentence pairs with high possibility of being context and answer. This is

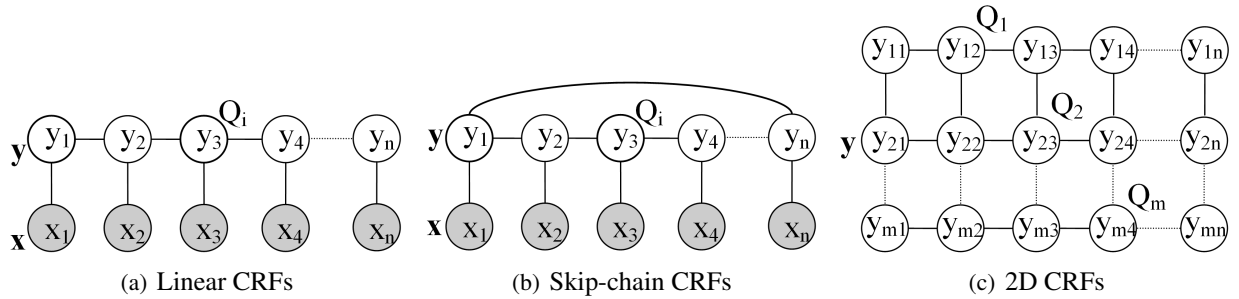


Figure 2: CRF Models

Skip-Chain	$y_v = A$	$y_v \neq A$
$y_u = C$	4,105	5,314
$y_u \neq C$	3,744	9,740

Table 2: Contingence table( $\chi^2=615.8, p\text{-value} < 0.001$ )

achieved as follows. Given a question  $Q_i$  in post  $P_j$  of a thread with  $n$  posts, its contexts usually occur within post  $P_j$  or before  $P_j$  while answers appear in the posts after  $P_j$ . We will establish an edge between each candidate answer  $v$  and one candidate context in  $\{P_k\}_{k=1}^j$  such that they have the highest possibility of being a context-answer pair of question  $Q_i$ :

$$u = \underset{u \in \{P_k\}_{k=1}^j}{\operatorname{argmax}} \operatorname{sim}(x_u, Q_i) \cdot \operatorname{sim}(x_v, \{x_u, Q_i\})$$

here, we use the product of  $\operatorname{sim}(x_u, Q_i)$  and  $\operatorname{sim}(x_v, \{x_u, Q_i\})$  to estimate the possibility of being a context-answer pair for  $(u, v)$ , where  $\operatorname{sim}(\cdot, \cdot)$  is the semantic similarity calculated on WordNet as described in Section 3.5. Table 2 shows that  $y_u$  and  $y_v$  in the skip chain generated by our heuristics influence each other significantly.

Skip-chain CRFs improve the performance of answer detection due to the introduced skip-chain edges that represent the joint probability conditioned on the question, which is exploited by skip-chain feature function:  $f(y_u, y_v, Q_i, \mathbf{x})$ .

### 3.3 Using 2D CRF Model

Both Linear CRFs and Skip-chain CRFs label the contexts and answers for each question in separate passes by assuming that questions in a thread are independent. Actually the assumption does not hold in many cases. Let us look at an example. As in Figure 1, sentence S10 is an answer for both question Q2 and Q3. S10 could be recognized as the answer of Q2 due to the shared word *areas* and *Causeway*

bay (in Q2’s context, S4), but there is no direct relation between Q3 and S10. To label S10, we need consider the dependency relation between Q2 and Q3. In other words, the question-answer relation between Q3 and S10 can be captured by a joint modeling of the dependency among S10, Q2 and Q3. The labels of the same sentence for two contiguous questions in a thread would be conditioned on the dependency relationship between the questions. Such a dependency cannot be captured by both Linear CRFs and Skip-chain CRFs.

To capture the dependency between the contiguous questions, we employ 2D CRFs to help context and answer detection. 2D CRF model is used in (Zhu et al., 2005) to model the neighborhood dependency in blocks within a web page. As shown in Figure2(c), 2D CRF models the labeling task for all questions in a thread. For each thread, there are  $m$  rows in the grid, where the  $i$ th row corresponds to one pass of Linear CRF model (or Skip-chain model) which labels contexts and answers for question  $Q_i$ . The vertical edges in the figure represent the joint probability conditioned on the contiguous questions, which will be exploited by 2D feature function:  $f(y_{i,j}, y_{i+1,j}, Q_i, Q_{i+1}, \mathbf{x})$ . Thus, the information generated in single CRF chain could be propagated over the whole grid. In this way, context and answer detection for all questions in the thread could be modeled together.

### 3.4 Conditional Random Fields (CRFs)

The Linear, Skip-Chain and 2D CRFs can be generalized as pairwise CRFs, which have two kinds of cliques in graph  $G$ : 1) node  $y_t$  and 2) edge  $(y_u, y_v)$ . The joint probability is defined as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{k,t} \lambda_k f_k(y_t, \mathbf{x}) + \sum_{k,t} \mu_k g_k(y_u, y_v, \mathbf{x})\right\}$$

where  $Z(\mathbf{x})$  is the normalization factor,  $f_k$  is the feature on nodes,  $g_k$  is on edges between  $u$  and  $v$ , and  $\lambda_k$  and  $\mu_k$  are parameters.

Linear CRFs are based on the first order Markov assumption that the contiguous nodes are dependent. The pairwise edges in Skip-chain CRFs represent the long distance dependency between the skipped nodes, while the ones in 2D CRFs represent the dependency between the neighboring nodes.

**Inference and Parameter Estimation.** For Linear CRFs, dynamic programming is used to compute the *maximum a posteriori* (MAP) of  $\mathbf{y}$  given  $\mathbf{x}$ . However, for more complicated graphs with cycles, exact inference needs the junction tree representation of the original graph and the algorithm is exponential to the treewidth. For fast inference, loopy Belief Propagation (Pearl, 1988) is implemented.

Given the training Data  $D = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^n$ , the parameter estimation is to determine the parameters based on maximizing the log-likelihood  $L_\lambda = \sum_{i=1}^n \log p(\mathbf{y}^{(i)} | \mathbf{x}^{(i)})$ . In Linear CRF model, dynamic programming and L-BFGS (limited memory Broyden-Fletcher-Goldfarb-Shanno) can be used to optimize objective function  $L_\lambda$ , while for complicated CRFs, Loopy BP are used instead to calculate the marginal probability.

### 3.5 Features used in CRF models

The main features used in Linear CRF models for context detection are listed in Table 3.

The similarity feature is to capture the word similarity and semantic similarity between candidate contexts and answers. The word similarity is based on cosine similarity of TF/IDF weighted vectors. The semantic similarity between words is computed based on Wu and Palmer’s measure (Wu and Palmer, 1994) using WordNet (Fellbaum, 1998).<sup>1</sup> The similarity between contiguous sentences will be used to capture the dependency for CRFs. In addition, to bridge the lexical gaps between question and context, we learned top-3 context terms for each question term from 300,000 question-description pairs obtained from Yahoo! Answers using mutual information (Berger et al., 2000) (question description in Yahoo! Answers is comparable to contexts in fo-

<sup>1</sup>The semantic similarity between sentences is calculated as in (Yang et al., 2006).

<p><u>Similarity features:</u></p> <ul style="list-style-type: none"> <li>· Cosine similarity with the question</li> <li>· Similarity with the question using WordNet</li> <li>· Cosine similarity between contiguous sentences</li> <li>· Similarity between contiguous sentences using WordNet</li> <li>· Cosine similarity with the expanded question using the lexical matching words</li> </ul> <p><u>Structural features:</u></p> <ul style="list-style-type: none"> <li>· The relative position to current question</li> <li>· Is its author the same with that of the question?</li> <li>· Is it in the same paragraph with its previous sentence?</li> </ul> <p><u>Discourse and lexical features:</u></p> <ul style="list-style-type: none"> <li>· The number of Pronouns in the question</li> <li>· The presence of fillers, fluency devices (e.g. “uh”, “ok”)</li> <li>· The presence of acknowledgment tokens</li> <li>· The number of non-stopwords</li> <li>· Whether the question has a noun or not?</li> <li>· Whether the question has a verb or not?</li> </ul>
--

Table 3: Features for Linear CRFs. Unless otherwise mentioned, we refer to features of the sentence whose label to be predicted

ums), and then use them to expand question and compute cosine similarity.

The structural features of forums provide strong clues for contexts. For example, contexts of a question usually occur in the post containing the question or preceding posts.

We extracted the discourse features from a question, such as the number of pronouns in the question. A more useful feature would be to find the entity in surrounding sentences referred by a pronoun. We tried GATE (Cunningham et al., 2002) for anaphora resolution of the pronouns in questions, but the performance became worse with the feature, which is probably due to the difficulty of anaphora resolution in forum discourse. We also observed that questions often need context if the question do not contain a noun or a verb.

In addition, we use similarity features between skip-chain sentences for Skip-chain CRFs and similarity features between questions for 2D CRFs.

## 4 Experiments

### 4.1 Experimental setup

**Corpus.** We obtained about 1 million threads from TripAdvisor forum; we randomly selected 591 threads and removed 22 threads which has more than 40 sentences and 6 questions; the remaining 579 forum threads form our corpus<sup>2</sup>. Each thread in our

<sup>2</sup>Tripadvisor (<http://www.tripadvisor.com/ForumHome>) is one of the most popular travel forums; the list of 579 urls is

Model	Prec(%)	Rec(%)	F <sub>1</sub> (%)
Context Detection			
SVM	75.27	68.80	71.32
C4.5	70.16	64.30	67.21
L-CRF	<b>75.75</b>	<b>72.84</b>	<b>74.45</b>
Answer Detection			
SVM	<b>73.31</b>	47.35	57.52
C4.5	65.36	46.55	54.37
L-CRF	63.92	<b>58.74</b>	<b>61.22</b>

Table 4: Context and Answer Detection

corpus contains at least two posts and on average each thread consists of 3.87 posts. Two annotators were asked to tag questions, their contexts, and answers in each thread. The kappa statistic for identifying question is 0.96, for linking context and question given a question is 0.75, and for linking answer and question given a question is 0.69. We conducted experiments on both the union and intersection of the two annotated data. The experimental results on both data are qualitatively comparable. We only report results on union data due to space limitation. The union data contains 1,064 questions, 1,458 contexts and 3,534 answers.

**Metrics.** We calculated precision, recall, and F<sub>1</sub>-score for all tasks. All the experimental results are obtained through the average of 5 trials of 5-fold cross validation.

## 4.2 Experimental results

### Linear CRFs for Context and Answer Detection.

This experiment is to evaluate Linear CRF model (Section 3.1) for context and answer detection by comparing with SVM and C4.5(Quinlan, 1993). For SVM, we use SVM<sup>light</sup>(Joachims, 1999). We tried linear, polynomial and RBF kernels and report the results on polynomial kernel using default parameters since it performs the best in the experiment. SVM and C4.5 use the same set of features as Linear CRFs. As shown in Table 4, Linear CRF model outperforms SVM and C4.5 for both context and answer detection. The main reason for the improvement is that CRF models can capture the sequential dependency between segments in forums as discussed in Section 3.1.

given in <http://homepages.inf.ed.ac.uk/gcong/ac108/>; Removing the 22 long threads can greatly reduce the training and test time.

position	Prec(%)	Rec(%)	F <sub>1</sub> (%)
Context Detection			
Previous One	63.69	34.29	44.58
Previous All	43.48	76.41	55.42
Answer Detection			
Following One	66.48	19.98	30.72
Following All	31.99	100	48.48

Table 5: Using position information for detection

Context	Prec(%)	Rec(%)	F <sub>1</sub> (%)
No context	63.92	58.74	61.22
Prev. sentence	61.41	62.50	61.84
Real context	63.54	66.40	64.94
L-CRF+context	65.51	63.13	64.06

Table 6: Contextual Information for Answer Detection. Prev. sentence uses one previous sentence of the current question as context. RealContext uses the context annotated by experts. L-CRF+context uses the context found by Linear CRFs

We next report a baseline of context detection using previous sentences in the same post with its question since contexts often occur in the question post or preceding posts. Similarly, we report a baseline of answer detecting using following segments of a question as answers. The results given in Table 5 show that location information is far from adequate to detect contexts and answers.

**The usefulness of contexts.** This experiment is to evaluate the usefulness of contexts in answer detection, by adding the similarity between the context (obtained with different methods) and candidate answer as an extra feature for CRFs. Table 6 shows the impact of context on answer detection using Linear CRFs. Linear CRFs with contextual information perform better than those without context. L-CRF+context is close to that using real context, while it is better than CRFs using the previous sentence as context. The results clearly shows that contextual information greatly improves the performance of answer detection.

**Improved Models.** This experiment is to evaluate the effectiveness of Skip-Chain CRFs (Section 3.2) and 2D CRFs (Section 3.3) for our tasks. The results are given in Table 7 and Table 8.

In context detection, Skip-Chain CRFs have simi-



Model	Prec(%)	Rec(%)	F <sub>1</sub> (%)
L-CRF+Context	75.75	72.84	74.45
Skip-chain	74.18	74.90	74.42
2D	75.92	76.54	76.41
2D+Skip-chain	<b>76.27</b>	<b>78.25</b>	<b>77.34</b>

Table 7: Skip-chain and 2D CRFs for context detection

lar results as Linear CRFs, i.e. the inter-dependency captured by the skip chains generated using the heuristics in Section 3.2 does not improve the context detection. The performance of Linear CRFs is improved in 2D CRFs (by 2%) and 2D+Skip-chain CRFs (by 3%) since they capture the dependency between contiguous questions.

In answer detection, as expected, Skip-chain CRFs outperform L-CRF+context since Skip-chain CRFs can model the inter-dependency between contexts and answers while in L-CRF+context the context can only be reflected by the features on the observations. We also observed that 2D CRFs improve the performance of L-CRF+context due to the dependency between contiguous questions. In contrast with our expectation, the 2D+Skip-chain CRFs does not improve Skip-chain CRFs in terms of answer detection. The possible reason could be that the structure of the graph is very complicated and too many parameters need to be learned on our training data.

**Evaluating Features.** We also evaluated the contributions of each category of features in Table 3 to context detection. We found that similarity features are the most important and structural feature the next. We also observed the same trend for answer detection. We omit the details here due to space limitation.

As a summary, 1) our CRF model outperforms SVM and C4.5 for both context and answer detections; 2) context is very useful in answer detection; 3) the Skip-chain CRF method is effective in leveraging context for answer detection; and 4) 2D CRF model improves the performance of Linear CRFs for both context and answer detection.

## 5 Discussions and Conclusions

We presented a new approach to detecting contexts and answers for questions in forums with good performance. We next discuss our experience not covered by the experiments, and future work.

Model	Prec(%)	Rec(%)	F <sub>1</sub> (%)
L-CRF+context	65.51	63.13	64.06
Skip-chain	<b>67.59</b>	<b>71.06</b>	<b>69.40</b>
2D	65.77	68.17	67.34
2D+Skip-chain	66.90	70.56	68.89

Table 8: Skip-chain and 2D CRFs for answer detection

Since contexts of questions are largely unexplored in previous work, we analyze the contexts in our corpus and classify them into three categories: 1) context contains the main content of question while question contains no constraint, e.g. “*i will visit NY at Oct, looking for a cheap hotel but convenient. Any good suggestion?* ”; 2) contexts explain or clarify part of the question, such as a definite noun phrase, e.g. “*We are going on the Taste of Paris. Does anyone know if it is advisable to take a suitcase with us on the tour.*”, where the first sentence is to describe *the tour*; and 3) contexts provide constraint or background for question that is syntactically complete, e.g. “*We are interested in visiting the Great Wall(and flying from London). Can anyone recommend a tour operator.*” In our corpus, about 26% questions do not need context, 12% questions need Type 1 context, 32% need Type 2 context and 30% Type 3. We found that our techniques often do not perform well on Type 3 questions.

We observed that factoid questions, one of focuses in the TREC QA community, take less than 10% question in our corpus. It would be interesting to revisit QA techniques to process forum data.

Other future work includes: 1) to summarize multiple threads using the triples extracted from individual threads. This could be done by clustering question-context-answer triples; 2) to use the traditional text summarization techniques to summarize the multiple answer segments; 3) to integrate the Question Answering techniques as features of our framework to further improve answer finding; 4) to reformulate questions using its context to generate more user-friendly questions for CQA services; and 5) to evaluate our techniques on more online forums in various domains.

## Acknowledgments

We thank the anonymous reviewers for their detailed comments, and Ming Zhou and Young-In Song for their valuable suggestions in preparing the paper.

## References

- A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. 2000. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of SIGIR*.
- J. Burger, C. Cardie, V. Chaudhri, R. Gaizauskas, S. Harabagiu, D. Israel, C. Jacquemin, C. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees, and R. Weischedel. 2006. Issues, tasks and program structures to roadmap research in question and answering (qna). ARAD: Advanced Research and Development Activity (US).
- G. Carenini, R. Ng, and X. Zhou. 2007. Summarizing email conversations with clue words. In *Proceedings of WWW*.
- G. Cong, L. Wang, C.Y. Lin, Y.I. Song, and Y. Sun. 2008. Finding question-answer pairs from online forums. In *Proceedings of SIGIR*.
- H. Cui, R. Sun, K. Li, M. Kan, and T. Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR*.
- H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of ACL*.
- H. Dang, J. Lin, and D. Kelly. 2007. Overview of the trec 2007 question answering track. In *Proceedings of TREC*.
- C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May.
- D. Feng, E. Shaw, J. Kim, and E. Hovy. 2006a. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proceedings of IUI*.
- D. Feng, E. Shaw, J. Kim, and E. Hovy. 2006b. Learning to detect conversation focus of threaded discussions. In *Proceedings of HLT-NAACL*.
- M. Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of EMNLP*.
- S. Harabagiu and A. Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of ACL*.
- J. Huang, M. Zhou, and D. Yang. 2007. Extracting chatbot knowledge from online discussion forums. In *Proceedings of IJCAI*.
- J. Jeon, W. Croft, and J. Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of CIKM*.
- T. Joachims. 1999. Making large-scale support vector machine learning practical. MIT Press, Cambridge, MA, USA.
- J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.
- A. McCallum and W. Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL-2003*.
- A. Nenkova and A. Bagga. 2003. Facilitating email thread access by extractive summary generation. In *Proceedings of RANLP*.
- J. Pearl. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- J. Quinlan. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- O. Rambow, L. Shrestha, J. Chen, and C. Lauridsen. 2004. Summarizing email threads. In *Proceedings of HLT-NAACL*.
- F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL*.
- L. Shrestha and K. McKeown. 2004. Detection of question-answer pairs in email conversations. In *Proceedings of COLING*.
- R. Soricut and E. Brill. 2006. Automatic question answering using the web: Beyond the Factoid. *Information Retrieval*, 9(2):191–206.
- C. Sutton and A. McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press. To appear.
- S. Wan and K. McKeown. 2004. Generating overview summaries of ongoing email thread discussions. In *Proceedings of COLING*.
- Z. Wu and M. S. Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of ACL*.
- F. Yang, J. Feng, and G. Fabbrizio. 2006. A data driven approach to relevancy recognition for contextual question answering. In *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL 2006*.
- L. Zhou and E. Hovy. 2005. Digesting virtual "geek" culture: The summarization of technical internet relay chats. In *Proceedings of ACL*.
- J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. 2005. 2d conditional random fields for web information extraction. In *Proceedings of ICML*.

# Learning to Rank Answers on Large Online QA Collections

Mihai Surdeanu, Massimiliano Ciaramita, Hugo Zaragoza

Barcelona Media Innovation Center, Yahoo! Research Barcelona

mihai.surdeanu@barcelonamedia.org, {massi, hugo}@yahoo-inc.com

## Abstract

This work describes an answer ranking engine for non-factoid questions built using a large online community-generated question-answer collection (Yahoo! Answers). We show how such collections may be used to effectively set up large supervised learning experiments. Furthermore we investigate a wide range of feature types, some exploiting NLP processors, and demonstrate that using them in combination leads to considerable improvements in accuracy.

## 1 Introduction

The problem of Question Answering (QA) has received considerable attention in the past few years. Nevertheless, most of the work has focused on the task of factoid QA, where questions match short answers, usually in the form of named or numerical entities. Thanks to international evaluations organized by conferences such as the Text REtrieval Conference (TREC)<sup>1</sup> or the Cross Language Evaluation Forum (CLEF) Workshop<sup>2</sup>, annotated corpora of questions and answers have become available for several languages, which has facilitated the development of robust machine learning models for the task.

The situation is different once one moves beyond the task of factoid QA. Comparatively little research has focused on QA models for non-factoid questions such as causation, manner, or reason questions. Because virtually no training data is available for this problem, most automated systems train either

<sup>1</sup><http://trec.nist.gov>

<sup>2</sup><http://www.clef-campaign.org>

High Quality	<i>Q</i> : How do you quiet a squeaky door? <i>A</i> : Spray WD-40 directly onto the hinges of the door. Open and close the door several times. Remove hinges if the door still squeaks. Remove any rust, dirt or loose paint. Apply WD-40 to removed hinges. Put the hinges back, open and close door several times again.
Low Quality	<i>Q</i> : How to extract html tags from an html documents with c++? <i>A</i> : very carefully

Table 1: Sample content from Yahoo! Answers.

on small hand-annotated corpora built in house (Higashinaka and Isozaki, 2008) or on question-answer pairs harvested from Frequently Asked Questions (FAQ) lists or similar resources (Soricut and Brill, 2006). None of these situations is ideal: the cost of building the training corpus in the former setup is high; in the latter scenario the data tends to be domain-specific, hence unsuitable for the learning of open-domain models.

On the other hand, recent years have seen an explosion of user-generated content (or social media). Of particular interest in our context are community-driven question-answering sites, such as Yahoo! Answers<sup>3</sup>, where users answer questions posed by other users and best answers are selected manually either by the asker or by all the participants in the thread. The data generated by these sites has significant advantages over other web resources: (a) it has a high growth rate and it is already abundant; (b) it covers a large number of topics, hence it offers a better

<sup>3</sup><http://answers.yahoo.com>

approximation of open-domain content; and (c) it is available for many languages. Community QA sites, similar to FAQs, provide large number of question-answer pairs. Nevertheless, this data has a significant drawback: it has high variance of quality, i.e., answers range from very informative to completely irrelevant or even abusive. Table 1 shows some examples of both high and low quality content.

In this paper we address the problem of answer ranking for non-factoid questions from social media content. Our research objectives focus on answering the following two questions:

1. *Is it possible to learn an answer ranking model for complex questions from such noisy data?* This is an interesting question because a positive answer indicates that a plethora of training data is readily available to QA researchers and system developers.
2. *Which features are most useful in this scenario?* Are similarity models as effective as models that learn question-to-answer transformations? Does syntactic and semantic information help? For generality, we focus only on textual features extracted from the answer text and we ignore all meta data information that is not generally available.

Notice that we concentrate on one component of a possible social-media QA system. In addition to answer ranking, a complete system would have to search for similar questions already answered (Jeon et al., 2005), and rank content quality using "social" features such as the authority of users (Jeon et al., 2006; Agichtein et al., 2008). This is not the focus of our work: here we investigate the problem of learning an answer ranking model capable of dealing with complex questions, using a large number of, possible noisy, question-answer pairs. By focusing exclusively on textual content we increase the portability of our approach to other collections where "social" features might not be available, e.g., Web search.

The paper is organized as follows. We describe our approach, including all the features explored for answer modeling, in Section 2. We introduce the corpus used in our empirical analysis in Section 3. We detail our experiments and analyze the results in Section 4. We overview related work in Section 5 and conclude the paper in Section 6.

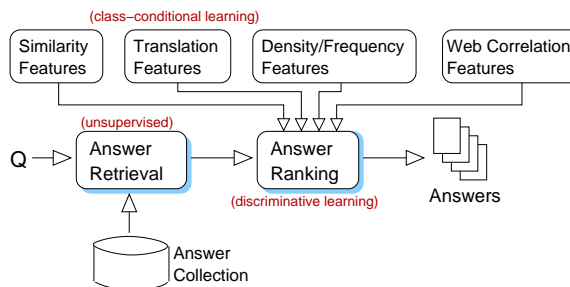


Figure 1: System architecture.

## 2 Approach

The architecture of the QA system analyzed in the paper, summarized in Figure 1, follows that of the most successful TREC systems. The first component, answer retrieval, extracts a set of candidate answers  $\mathbf{A}$  for question  $Q$  from a large collection of answers,  $\mathbf{C}$ , provided by a community-generated question-answering site. The retrieval component uses a state-of-the-art information retrieval (IR) model to extract  $\mathbf{A}$  given  $Q$ . Since our focus is on exploring the usability of the answer content, we do not perform retrieval by finding similar questions already answered (Jeon et al., 2005), i.e., our answer collection  $\mathbf{C}$  contains only the site’s answers without the corresponding questions answered.

The second component, answer ranking, assigns to each answer  $A_i \in \mathbf{A}$  a score that represents the likelihood that  $A_i$  is a correct answer for  $Q$ , and ranks all answers in descending order of these scores. The scoring function is a linear combination of four different classes of features (detailed in Section 2.2). This function is the focus of the paper. To answer our first research objective we will compare the quality of the rankings provided by this component against the rankings generated by the IR model used for answer retrieval. To answer the second research objective we will analyze the contribution of the proposed feature set to this function. Again, since our interest is in investigating the utility of the answer textual content, we use only information extracted from the answer text when learning the scoring function. We do not use any meta information (e.g., answerer credibility, click counts, etc.) (Agichtein et al., 2008; Jeon et al., 2006).

Our QA approach combines three types of machine learning methodologies (as highlighted in Figure 1): the answer retrieval component uses un-

supervised IR models, the answer ranking is implemented using discriminative learning, and finally, some of the ranking features are produced by question-to-answer translation models, which use class-conditional learning.

## 2.1 Ranking Model

Learning with user-generated content can involve arbitrarily large amounts of data. For this reason we choose as a ranking algorithm the Perceptron which is both accurate and efficient and can be trained with online protocols. Specifically, we implement the ranking Perceptron proposed by Shen and Joshi (2005), which reduces the ranking problem to a binary classification problem. The general intuition is to exploit the pairwise preferences induced from the data by training on pairs of patterns, rather than independently on each pattern. Given a weight vector  $\alpha$ , the score for a pattern  $\mathbf{x}$  (a candidate answer) is simply the inner product between the pattern and the weight vector:

$$f_{\alpha}(\mathbf{x}) = \langle \mathbf{x}, \alpha \rangle \quad (1)$$

However, the error function depends on pairwise scores. In training, for each pair  $(\mathbf{x}_i, \mathbf{x}_j) \in \mathbf{A}$ , the score  $f_{\alpha}(\mathbf{x}_i - \mathbf{x}_j)$  is computed; note that if  $f$  is an inner product  $f_{\alpha}(\mathbf{x}_i - \mathbf{x}_j) = f_{\alpha}(\mathbf{x}_i) - f_{\alpha}(\mathbf{x}_j)$ . Given a margin function  $g(i, j)$  and a positive rate  $\tau$ , if  $f_{\alpha}(\mathbf{x}_i - \mathbf{x}_j) \leq g(i, j)\tau$ , an update is performed:

$$\alpha^{t+1} = \alpha^t + (\mathbf{x}_i - \mathbf{x}_j)\tau g(i, j) \quad (2)$$

By default we use  $g(i, j) = (\frac{1}{i} - \frac{1}{j})$ , as a margin function, as suggested in (Shen and Joshi, 2005), and find  $\tau$  empirically on development data. Given that there are only two possible ranks in our setting, this function only generates two possible values. For regularization purposes, we use as a final model the average of all Perceptron models posited during training (Freund and Schapire, 1999).

## 2.2 Features

In the scoring model we explore a rich set of features inspired by several state-of-the-art QA systems. We investigate how such features can be adapted and combined for non-factoid answer ranking, and perform a comparative feature analysis using a significant amount of real-world data. For clarity, we group the features into four sets: features that model

the similarity between questions and answers (FG1), features that encode question-to-answer transformations using a translation model (FG2), features that measure keyword density and frequency (FG3), and features that measure the correlation between question-answer pairs and other collections (FG4). Wherever applicable, we explore different syntactic and semantic representations of the textual content, e.g., extracting the dependency-based representation of the text or generalizing words to their WordNet supersenses (WNSS) (Ciaramita and Altun, 2006). We detail each of these feature groups next.

### FG1: Similarity Features

We measure the similarity between a question  $Q$  and an answer  $A$  using the length-normalized *BM25* formula (Robertson and Walker, 1997). We chose this similarity formula because, out of all the IR models we tried, it provided the best ranking at the output of the answer retrieval component. For completeness we also include in the feature set the value of the *tf · idf* similarity measure. For both formulas we use the implementations available in the Terrier IR platform<sup>4</sup> with the default parameters.

To understand the contribution of our syntactic and semantic processors we compute the above similarity features for five different representations of the question and answer content:

*Words (W)* - this is the traditional IR view where the text is seen as a bag of words.

*Dependencies (D)* - the text is represented as a bag of binary syntactic dependencies. The relative syntactic processor is detailed in Section 3. Dependencies are fully lexicalized but unlabeled and we currently extract dependency paths of length 1, i.e., direct head-modifier relations (this setup achieved the best performance).

*Generalized dependencies (D<sub>g</sub>)* - same as above, but the words in dependencies are generalized to their WNSS, if detected.

*Bigrams (B)* - the text is represented as a bag of bigrams (larger  $n$ -grams did not help). We added this view for a fair analysis of the above syntactic views.

*Generalized bigrams (B<sub>g</sub>)* - same as above, but the words are generalized to their WNSS.

<sup>4</sup><http://ir.dcs.gla.ac.uk/terrier>

In all these representations we skip stop words and normalize all words to their WordNet lemmas.

## FG2: Translation Features

Berger et al. (2000) showed that similarity-based models are doomed to perform poorly for QA because they fail to “bridge the lexical chasm” between questions and answers. One way to address this problem is to learn question-to-answer transformations using a translation model (Berger et al., 2000; Echihiabi and Marcu, 2003; Soricut and Brill, 2006; Riezler et al., 2007). In our model, we incorporate this approach by adding the probability that the question  $Q$  is a translation of the answer  $A$ ,  $P(Q|A)$ , as a feature. This probability is computed using IBM’s Model 1 (Brown et al., 1993):

$$P(Q|A) = \prod_{q \in Q} P(q|A) \quad (3)$$

$$P(q|A) = (1 - \lambda)P_{ml}(q|A) + \lambda P_{ml}(q|C) \quad (4)$$

$$P_{ml}(q|A) = \sum_{a \in A} (T(q|a)P_{ml}(a|A)) \quad (5)$$

where the probability that the question term  $q$  is generated from answer  $A$ ,  $P(q|A)$ , is smoothed using the prior probability that the term  $q$  is generated from the entire collection of answers  $C$ ,  $P_{ml}(q|C)$ .  $\lambda$  is the smoothing parameter.  $P_{ml}(q|C)$  is computed using the maximum likelihood estimator.  $P_{ml}(q|A)$  is computed as the sum of the probabilities that the question term  $q$  is a translation of an answer term  $a$ ,  $T(q|a)$ , weighted by the probability that  $a$  is generated from  $A$ . The translation table for  $T(q|a)$  is computed using the EM-based algorithm implemented in the GIZA++ toolkit<sup>5</sup>.

Similarly with the previous feature group, we add translation-based features for the five different text representations introduced above. By moving beyond the bag-of-words representation we hope to learn relevant transformations of structures, e.g., from the “squeaky”  $\rightarrow$  “door” dependency to “spray”  $\leftarrow$  “WD-40” in the Table 1 example.

## FG3: Density and Frequency Features

These features measure the density and frequency of question terms in the answer text. Variants of these features were used previously for either answer or passage ranking in factoid QA (Moldovan et al., 1999; Harabagiu et al., 2000).

<sup>5</sup><http://www.fjoch.com/GIZA++.html>

*Same word sequence* - computes the number of non-stop question words that are recognized in the same order in the answer.

*Answer span* - the largest distance (in words) between two non-stop question words in the answer.

*Same sentence match* - number of non-stop question terms matched in a single sentence in the answer.

*Overall match* - number of non-stop question terms matched in the complete answer.

These last two features are computed also for the other four text representations previously introduced ( $B$ ,  $B_g$ ,  $D$ , and  $D_g$ ). Counting the number of matched dependencies is essentially a simplified tree kernel for QA (e.g., see (Moschitti et al., 2007)) matching only trees of depth 2. Experiments with full dependency tree kernels based on several variants of the convolution kernels of Collins and Duffy (2001) did not yield improvements. We conjecture that the mistakes of the syntactic parser may be amplified in tree kernels, which consider an exponential number of sub-trees.

*Informativeness* - we model the amount of information contained in the answer by counting the number of non-stop nouns, verbs, and adjectives in the answer text that do not appear in the question.

## FG4: Web Correlation Features

Previous work has shown that the redundancy of a large collection (e.g., the web) can be used for answer validation (Brill et al., 2001; Magnini et al., 2002). In the same spirit, we add features that measure the correlation between question-answer pairs and large external collections:

*Web correlation* - we measure the correlation between the question-answer pair and the web using the Corrected Conditional Probability (CCP) formula of Magnini et al. (2002):  $CCP(Q, A) = hits(Q + A) / (hits(Q) hits(A)^{2/3})$  where *hits* returns the number of page hits from a search engine. When a query returns zero hits we iteratively relax it by dropping the keyword with the smallest priority. Keyword priorities are assigned using the heuristics of Moldovan et al. (1999).

*Query-log correlation* - as in (Ciaramita et al., 2008) we also compute the correlation between question-answer pairs and a search-engine query-log corpus of more than 7.5 million queries, which shares

roughly the same time stamp with the community-generated question-answer corpus. We compute the Pointwise Mutual Information (PMI) and Chi square ( $\chi^2$ ) association measures between each question-answer word pair in the query-log corpus. The largest and the average values are included as features, as well as the number of QA word pairs which appear in the top 10, 5, and 1 percentile of the PMI and  $\chi^2$  word pair rankings.

### 3 The Corpus

The corpus is extracted from a sample of the U.S. Yahoo! Answers logs. In this paper we focus on the subset of advice or “how to” questions due to their frequency and importance in social communities.<sup>6</sup> To construct our corpus, we implemented the following successive filtering steps:

Step 1: from the full corpus we keep only questions that match the regular expression:

```
how (to|do|did|does|can|would|could|should)
and have an answer selected as best either by
the asker or by the participants in the thread.
The outcome of this step is a set of 364,419
question-answer pairs.
```

Step 2: from the above corpus we remove the questions and answers of obvious low quality. We implement this filter with a simple heuristic by keeping only questions and answers that have at least 4 words each, out of which at least 1 is a noun and at least 1 is a verb. This step filters out questions like “How to be excellent?” and answers such as “I don’t know”. The outcome of this step forms our answer collection C. C contains 142,627 question-answer pairs.<sup>7</sup>

Arguably, all these filters could be improved. For example, the first step can be replaced by a question classifier (Li and Roth, 2005). Similarly, the second step can be implemented with a statistical classifier that ranks the quality of the content using both the textual and non-textual information available in the database (Jeon et al., 2006; Agichtein et al., 2008). We plan to further investigate these issues which are not the main object of this work.

<sup>6</sup>Nevertheless, the approach proposed here is independent of the question type. We will explore answer ranking for other non-factoid question types in future work.

<sup>7</sup>The data will be available through the Yahoo! Webscope program (research-data-requests@yahoo-inc.com).

The data was processed as follows. The text was split at the sentence level, tokenized and PoS tagged, in the style of the Wall Street Journal Penn Tree-Bank (Marcus et al., 1993). Each word was morphologically simplified using the morphological functions of the WordNet library<sup>8</sup>. Sentences were annotated with WNSS categories, using the tagger of Ciaramita and Altun (2006)<sup>9</sup>, which annotates text with a 46-label tagset. These tags, defined by WordNet lexicographers, provide a broad semantic categorization for nouns and verbs and include labels for nouns such as food, animal, body and feeling, and for verbs labels such as communication, contact, and possession. Next, we parsed all sentences with the dependency parser of Attardi et al. (2007)<sup>10</sup>. It is important to realize that the output of all mentioned processing steps is noisy and contains plenty of mistakes, since the data has huge variability in terms of quality, style, genres, domains etc., and domain adaptation for the NLP tasks involved is still an open problem (Dredze et al., 2007).

We used 60% of the questions for training, 20% for development, and 20% for test. The candidate answer set for a given question is composed by one positive example, i.e., its corresponding best answer, and as negative examples all the other answers retrieved in the top  $N$  by the retrieval component.

### 4 Experiments

We evaluate our results using two measures: mean Precision at rank=1 (P@1) – i.e., the percentage of questions with the correct answer on the first position – and Mean Reciprocal Rank (MRR) – i.e., the score of a question is  $1/k$ , where  $k$  is the position of the correct answer. We use as baseline the output of our answer retrieval component (Figure 1). This component uses the BM25 criterion, the highest performing IR model in our experiments.

Table 2 lists the results obtained using this baseline and our best model (“Ranking” in the table) on the testing partition. Since we are interested in the performance of the ranking model, we evaluate on the subset of questions where the correct answer is retrieved by answer retrieval in the top  $N$  answers (similar to Ko et al. (2007)). In the table we report

<sup>8</sup><http://wordnet.princeton.edu>

<sup>9</sup>[sourceforge.net/projects/supersensetag](http://sourceforge.net/projects/supersensetag)

<sup>10</sup><http://sourceforge.net/projects/desr>

	MRR				P@1			
	$N = 10$	$N = 15$	$N = 25$	$N = 50$	$N = 10$	$N = 15$	$N = 25$	$N = 50$
recall@N	26.25%	29.04%	32.81%	38.09%	26.25%	29.04%	32.81%	38.09%
Baseline	61.33	56.12	50.31	43.74	45.94	41.48	36.74	31.66
Ranking	<b>68.72</b> $\pm 0.01$	<b>63.84</b> $\pm 0.01$	<b>57.76</b> $\pm 0.07$	<b>50.72</b> $\pm 0.01$	<b>54.22</b> $\pm 0.01$	<b>49.59</b> $\pm 0.03$	<b>43.98</b> $\pm 0.09$	<b>37.99</b> $\pm 0.01$
Improvement	+12.04%	+13.75%	+14.80%	+15.95%	+18.02%	+19.55%	+19.70%	+19.99%

Table 2: Overall results for the test partition.

results for several  $N$  values. For completeness, we show the percentage of questions that match this criterion in the “recall@N” row.

Our ranking model was tuned strictly on the development set (i.e., feature selection and parameters of the translation models). During training, the presentation of the training instances is randomized, which generates a randomized ranking algorithm. We exploit this property to estimate the variance in the results produced by each model and report the average result over 10 trials together with an estimate of the standard deviation.

The baseline result shows that, for  $N = 15$ , BM25 alone can retrieve in first rank 41% of the correct answers, and MRR tells us that the correct answer is often found within the first three answers (this is not so surprising if we remember that in this configuration only questions with the correct answer in the first 15 were kept for the experiment). The baseline results are interesting because they indicate that the problem is not hopelessly hard, but it is far from trivial. In principle, we see much room for improvement over bag-of-word methods.

Next we see that learning a weighted combination of features yields consistently marked improvements: for example, for  $N = 15$ , the best model yields a 19% relative improvement in P@1 and 14% in MRR. More importantly, the results indicate that the model learned is stable: even though for the model analyzed in Table 2 we used  $N = 15$  in training, we measure approximately the same relative improvement as  $N$  increases during evaluation.

These results provide robust evidence that: (a) we can use publicly available online QA collections to investigate features for answer ranking without the need for costly human evaluation, (b) we can exploit large and noisy online QA collections to improve the accuracy of answer ranking systems and (c) readily available and scalable NLP technology can be used

Iter.	Feature Set	MRR	P@1
0	BM25(W)	56.06	41.12%
1	+ translation( $B_g$ )	61.13	46.24%
2	+ overall match(D)	62.50	48.34%
3	+ translation(W)	63.00	49.08%
4	+ query-log avg( $\chi^2$ )	63.50	49.63%
5	+ answer span normalized by A size	63.71	49.84%
6	+ query-log max(PMI)	63.87	50.09%
7	+ same word sequence	63.99	50.23%
8	+ translation(B)	64.03	50.30%
9	+ tfidf(W)	64.08	50.42%
10	+ same sentence match(W)	64.10	50.42%
11	+ informativeness: verb count	64.18	50.36%
12	+ tfidf(B)	64.22	50.36%
13	+ same word sequence normalized by Q size	64.33	50.54%
14	+ query-log max( $\chi^2$ )	64.46	50.66%
15	+ same sentence match(W) normalized by Q size	64.55	50.78%
16	+ query-log avg(PMI)	64.60	50.88%
17	+ overall match(W)	64.65	50.91%

Table 3: Summary of the model selection process.

to improve lexical matching and translation models. In the remaining of this section we analyze the performance of the different features.

Table 3 summarizes the outcome of our automatic greedy feature selection process on the development set. Where applicable, we show within parentheses the text representation for the corresponding feature. The process is initialized with a single feature that replicates the baseline model (BM25 applied to the bag-of-words (W) representation). The algorithm incrementally adds to the feature set the feature that provides the highest MRR improvement in the development partition. The process stops when no features yield any improvement. The table shows that, while the features selected span all the four feature groups introduced, the lion’s share is taken by the translation features: approximately 60% of the MRR



	W	B	$B_g$	D	$D_g$	W + B	W + B + $B_g$	W + B + $B_g$ + D	W + B + $B_g$ + D + $D_g$
FG1 (Similarity)	0	<b>+1.06</b>	-2.01	+0.84	-1.75	+1.06	+1.06	+1.06	+1.06
FG2 (Translation)	+4.95	+4.73	+5.06	+4.63	+4.66	+5.80	+6.01	<b>+6.36</b>	+6.36
FG3 (Frequency)	+2.24	+2.33	+2.39	+2.27	+2.41	+3.56	+3.56	<b>+3.62</b>	+3.62

Table 4: Contribution of NLP processors. Scores are MRR improvements on the development set.

improvement is achieved by these features. The frequency/density features are responsible for approximately 23% of the improvement. The rest is due to the query-log correlation features. This indicates that, even though translation models are the most useful, it is worth exploring approaches that combine several strategies for answer ranking.

Note that if some features do not appear in Table 3 it does not necessarily mean that they are useless. In some cases such features are highly correlated with features previously selected, which already exploited their signal. For example, most similarity features (FG1) are correlated. Because BM25(W) is part of the baseline model, the selection process chooses another FG1 feature only much later (iteration 9) when the model is significantly changed. On the other hand, some features do not provide a useful signal at all. A notable example in this class is the web-based CCP feature, which was designed originally for factoid answer validation and does not adapt well to our problem. Because the length of non-factoid answers is typically significantly larger than in the factoid QA task, we have to discard a large part of the query when computing  $hits(Q+A)$  to reach non-zero counts. This means that the final hit counts, hence the CCP value, are generally uncorrelated with the original (Q,A) tuple.

One interesting observation is that the first two features chosen by our model selection process use information from the NLP processors. The first chosen feature is the translation probability computed between the  $B_g$  question and answer representations (bigrams with words generalized to their WNSS tags). The second feature selected measures the number of syntactic dependencies from the question that are matched in the answer. These results provide empirical evidence that coarse semantic disambiguation and syntactic parsing have a positive contribution to non-factoid QA, even in broad-coverage noisy settings based on Web data.

The above observation deserves a more detailed

analysis. Table 4 shows the performance of our first three feature groups when they are applied to each of the five text representations or incremental combinations of representations. For each model corresponding to a table cell we use only the features from the corresponding feature group and representation to avoid the correlation with features from other groups. We generate each best model using the same feature selection process described above.

The left part of Table 4 shows that, generally, the models using representations that include the output of our NLP processors ( $B_g$ , D and  $D_g$ ) improve over the baseline (FG1 and W).<sup>11</sup> However, comparable improvements can be obtained with the simpler bigram representation (B). This indicates that, in terms of individual contributions, our NLP processors can be approximated with simpler  $n$ -gram models in this task. Hence, is it fair to say that syntactic and semantic analysis is useful for such Web QA tasks? While the above analysis seems to suggest a negative answer, the right-hand side of Table 4 tells a more interesting story. It shows that the NLP analysis provides *complementary* information to the  $n$ -gram-based models. The best models for the FG2 and FG3 feature groups are obtained when combining the  $n$ -gram representations with the representations that use the output of the NLP processors (W + B +  $B_g$  + D). The improvements are relatively small, but remarkable (e.g., see FG2) if we take into account the significant scale of the evaluation. This observation correlates well with the analysis shown in Table 3, which shows that features using semantic ( $B_g$ ) and syntactic (D) representations contribute the most *on top* of the IR model (BM25(W)).

<sup>11</sup>The exception to this rule are the models FG1( $B_g$ ) and FG1( $D_g$ ). This is caused by the fact that the BM25 formula is less forgiving with errors of the NLP processors (due to the high *idf* scores assigned to bigrams and dependencies), and the WNSS tagger is the least robust component in our pipeline.

## 5 Related Work

Content from community-built question-answer sites can be retrieved by searching for similar questions already answered (Jeon et al., 2005) and ranked using meta-data information like answerer authority (Jeon et al., 2006; Agichtein et al., 2008). Here we show that the answer text can be successfully used to improve answer ranking quality. Our method is complementary to the above approaches. In fact, it is likely that an optimal retrieval engine from social media should combine all these three methodologies. Moreover, our approach might have applications outside of social media (e.g., for open-domain web-based QA), because the ranking model built is based only on open-domain knowledge and the analysis of textual content.

In the QA literature, answer ranking for non-factoid questions has typically been performed by learning question-to-answer transformations, either using translation models (Berger et al., 2000; Soricut and Brill, 2006) or by exploiting the redundancy of the Web (Agichtein et al., 2001). Girju (2003) extracts non-factoid answers by searching for certain semantic structures, e.g., causation relations as answers to causation questions. In this paper we combine several methodologies, including the above, into a single model. This approach allowed us to perform a systematic feature analysis on a large-scale real-world corpus and a comprehensive feature set.

Recent work has showed that structured retrieval improves answer ranking for factoid questions: Bilotti et al. (2007) showed that matching predicate-argument frames constructed from the question and the expected answer types improves answer ranking. Cui et al. (2005) learned transformations of dependency paths from questions to answers to improve passage ranking. However, both approaches use similarity models at their core because they require the matching of the lexical elements in the search structures. On the other hand, our approach allows the learning of full transformations from question structures to answer structures using translation models applied to different text representations.

Our answer ranking framework is closest in spirit to the system of Ko et al. (2007) or Higashinaka et al. (2008). However, the former was applied only to factoid QA and both are limited to similarity, re-

dundancy and gazetteer-based features. Our model uses a larger feature set that includes correlation and transformation-based features and five different content representations. Our evaluation is also carried out on a larger scale. Our work is also related to that of Riezler et al. (2007) where SMT-based query expansion methods are used on data from FAQ pages.

## 6 Conclusions

In this work we described an answer ranking engine for non-factoid questions built using a large community-generated question-answer collection. On one hand, this study shows that we can effectively exploit large amounts of available Web data to do research on NLP for non-factoid QA systems, without any annotation or evaluation cost. This provides an excellent framework for large-scale experimentation with various models that otherwise might be hard to understand or evaluate. On the other hand, we expect the outcome of this process to help several applications, such as open-domain QA on the Web and retrieval from social media. For example, on the Web our ranking system could be combined with a passage retrieval system to form a QA system for complex questions. On social media, our system should be combined with a component that searches for similar questions already answered; this output can possibly be filtered further by a content-quality module that explores “social” features such as the authority of users, etc.

We show that the best ranking performance is obtained when several strategies are combined into a single model. We obtain the best results when similarity models are aggregated with features that model question-to-answer transformations, frequency and density of content, and correlation of QA pairs with external collections. While the features that model question-to-answer transformations provide most benefits, we show that the combination is crucial for improvement.

Lastly, we show that syntactic dependency parsing and coarse semantic disambiguation yield a small, yet statistically significant performance increase on top of the traditional bag-of-words and  $n$ -gram representation. We obtain these results using only off-the-shelf NLP processors that were not adapted in any way for our task.

## References

- G. Attardi, F. Dell'Orletta, M. Simi, A. Chanev and M. Ciaramita. 2007. Multilingual Dependency Parsing and Domain Adaptation using DeSR. *Proc. of CoNLL Shared Task Session of EMNLP-CoNLL 2007*.
- E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. 2008. Finding High-Quality Content in Social Media, with an Application to Community-based Question Answering. *Proc. of WSDM*.
- E. Agichtein, S. Lawrence, and L. Gravano. 2001. Learning Search Engine Specific Query Transformations for Question Answering. *Proc. of WWW*.
- A. Berger, R. Caruana, D. Cohn, D. Freytag, and V. Mittal. 2000. Bridging the Lexical Chasm: Statistical Approaches to Answer Finding. *Proc. of SIGIR*.
- M. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. 2007. Structured Retrieval for Question Answering. *Proc. of SIGIR*.
- E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-Intensive Question Answering. *Proc. of TREC*.
- P. Brown, S. Della Pietra, V. Della Pietra, R. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2).
- M. Ciaramita and Y. Altun. 2006. Broad Coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger. *Proc. of EMNLP*.
- M. Ciaramita, V. Murdock and V. Plachouras. 2008. Semantic Associations for Contextual Advertising. 2008. *Journal of Electronic Commerce Research - Special Issue on Online Advertising and Sponsored Search*, 9(1), pp.1-15.
- M. Collins and N. Duffy. 2001. Convolution Kernels for Natural Language. *Proc. of NIPS 2001*.
- H. Cui, R. Sun, K. Li, M. Kan, and T. Chua. 2005. Question Answering Passage Retrieval Using Dependency Relations. *Proc. of SIGIR*.
- M. Dredze, J. Blitzer, P. Pratim Talukdar, K. Ganchev, J. Graca, and F. Pereira. 2007. Frustratingly Hard Domain Adaptation for Parsing. In *Proc. of EMNLP-CoNLL 2007 Shared Task*.
- A. Echihabi and D. Marcu. 2003. A Noisy-Channel Approach to Question Answering. *Proc. of ACL*.
- Y. Freund and R.E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37, pp. 277-296.
- R. Girju. 2003. Automatic Detection of Causal Relations for Question Answering. *Proc. of ACL, Workshop on Multilingual Summarization and Question Answering*.
- S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. 2000. Falcon: Boosting Knowledge for Answer Engines. *Proc. of TREC*.
- R. Higashinaka and H. Isozaki. 2008. Corpus-based Question Answering for why-Questions. *Proc. of IJCNLP*.
- J. Jeon, W. B. Croft, and J. H. Lee. 2005. Finding Similar Questions in Large Question and Answer Archives. *Proc. of CIKM*.
- J. Jeon, W. B. Croft, J. H. Lee, and S. Park. 2006. A Framework to Predict the Quality of Answers with Non-Textual Features. *Proc. of SIGIR*.
- J. Ko, T. Mitamura, and E. Nyberg. 2007. Language-independent Probabilistic Answer Ranking. for Question Answering. *Proc. of ACL*.
- X. Li and D. Roth. 2005. Learning Question Classifiers: The Role of Semantic Information. *Natural Language Engineering*.
- B. Magnini, M. Negri, R. Prevete, and H. Tanev. 2002. Comparing Statistical and Content-Based Techniques for Answer Validation on the Web. *Proc. of the VIII Convegno AI\*IA*.
- M.P. Marcus, B. Santorini and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn TreeBank. *Computational Linguistics*, 19(2), pp. 313-330.
- D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, and V. Rus. 1999. LASSO - A Tool for Surfing the Answer Net. *Proc. of TREC*.
- A. Moschitti, S. Quarteroni, R. Basili and S. Manandhar. 2007. Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classification. *Proc. of ACL*.
- S. Robertson and S. Walker. 1997. On relevance Weights with Little Relevance Information. *Proc. of SIGIR*.
- R. Soricut and E. Brill. 2006. Automatic Question Answering Using the Web: Beyond the Factoid. *Journal of Information Retrieval - Special Issue on Web Information Retrieval*, 9(2).
- L. Shen and A. Joshi. 2005. Ranking and Reranking with Perceptron, *Machine Learning. Special Issue on Learning in Speech and Language Technologies*, 60(1-3), pp. 73-96.
- S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal and Y. Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proc. of ACL*.

# Unsupervised Lexicon-Based Resolution of Unknown Words for Full Morphological Analysis

Meni Adler and Yoav Goldberg and David Gabay and Michael Elhadad

Ben Gurion University of the Negev

Department of Computer Science\*

POB 653 Be'er Sheva, 84105, Israel

{adlerm, goldberg, gabayd, elhadad}@cs.bgu.ac.il

## Abstract

Morphological disambiguation proceeds in 2 stages: (1) an analyzer provides all possible analyses for a given token and (2) a stochastic disambiguation module picks the most likely analysis in context. When the analyzer does not recognize a given token, we hit the problem of unknowns. In large scale corpora, unknowns appear at a rate of 5 to 10% (depending on the genre and the maturity of the lexicon).

We address the task of computing the distribution  $p(t|w)$  for unknown words for full morphological disambiguation in Hebrew. We introduce a novel algorithm that is language independent: it exploits a maximum entropy letters model trained over the known words observed in the corpus and the distribution of the unknown words in known tag contexts, through iterative approximation. The algorithm achieves 30% error reduction on disambiguation of unknown words over a competitive baseline (to a level of 70% accurate full disambiguation of unknown words). We have also verified that taking advantage of a strong language-specific model of morphological patterns provides the same level of disambiguation. The algorithm we have developed exploits distributional information latent in a wide-coverage lexicon and large quantities of unlabeled data.

## 1 Introduction

The term *unknowns* denotes tokens in a text that cannot be resolved in a given lexicon. For the task of full morphological analysis, the lexicon must provide all possible morphological analyses for any given token. In this case, unknown tokens can be categorized into two classes of missing information: *unknown tokens* are not recognized at all by the lexicon, and *unknown analyses*, where the set of analyses for a lexeme does not contain the correct analysis for a given token. Despite efforts on improving the underlying lexicon, unknowns typically represent 5% to 10% of the number of tokens in large-scale corpora. The alternative to continuously investing manual effort in improving the lexicon is to design methods to learn possible analyses for unknowns from observable features: their letter structure and their context. In this paper, we investigate the characteristics of Hebrew unknowns for full morphological analysis, and propose a new method for handling such unavoidable lack of information. Our method generates a distribution of possible analyses for unknowns. In our evaluation, these learned distributions include the correct analysis for unknown words in 85% of the cases, contributing an error reduction of over 30% over a competitive baseline for the overall task of full morphological analysis in Hebrew.

The task of a morphological analyzer is to produce all possible analyses for a given token. In Hebrew, the analysis for each token is of the form lexeme-and-features<sup>1</sup>: lemma, affixes, lexical cate-

---

This work is supported in part by the Lynn and William Frankel Center for Computer Science.

---

<sup>1</sup>In contrast to the prefix-stem-suffix analysis format of

gory (POS), and a set of inflection properties (according to the POS) – gender, number, person, status and tense. In this work, we refer to the morphological analyzer of MILA – the Knowledge Center for Processing Hebrew<sup>2</sup> (hereafter *KC analyzer*). It is a synthetic analyzer, composed of two data resources – a lexicon of about 2,400 lexemes, and a set of generation rules (see (Adler, 2007, Section 4.2)). In addition, we use an unlabeled text corpus, composed of stories taken from three Hebrew daily news papers (Aruts 7, Haaretz, The Marker), of 42M tokens. We observed 3,561 different composite tags (*e.g.*, noun-sing-fem-prepPrefix:be) over this corpus. These 3,561 tags form the large tagset over which we train our learner. On the one hand, this tagset is much larger than the largest tagset used in English (from 17 tags in most unsupervised POS tagging experiments, to the 46 tags of the WSJ corpus and the about 150 tags of the LOB corpus). On the other hand, our tagset is intrinsically factored as a set of dependent sub-features, which we explicitly represent.

The task we address in this paper is morphological disambiguation: given a sentence, obtain the list of all possible analyses for each word from the analyzer, and disambiguate each word in context. On average, each token in the 42M corpus is given 2.7 possible analyses by the analyzer (much higher than the average 1.41 POS tag ambiguity reported in English (Dermatas and Kokkinakis, 1995)). In previous work, we report disambiguation rates of 89% for full morphological disambiguation (using an unsupervised EM-HMM model) and 92.5% for part of speech and segmentation (without assigning all the inflectional features of the words).

In order to estimate the importance of unknowns in Hebrew, we analyze tokens in several aspects: (1) the number of unknown tokens, as observed on the corpus of 42M tokens; (2) a manual classification of a sample of 10K unknown token types out of the 200K unknown types identified in the corpus; (3) the number of unknown analyses, based on an annotated corpus of 200K tokens, and their classification.

About 4.5% of the 42M token instances in the

Buckwalter’s Arabic analyzer (2004), which looks for any legal combination of prefix-stem-suffix, but does not provide full morphological features such as gender, number, case etc.

<sup>2</sup><http://mila.cs.technion.ac.il/html>

training corpus were unknown tokens (45% of the 450K token types). For less edited text, such as random text sampled from the Web, the percentage is much higher – about 7.5%. In order to classify these unknown tokens, we sampled 10K unknown token types and examined them manually. The classification of these tokens with their distribution is shown in Table 1<sup>3</sup>. As can be seen, there are two main classes of unknown token types: Neologisms (32%) and Proper nouns (48%), which cover about 80% of the unknown token instances. The POS distribution of the unknown tokens of our annotated corpus is shown in Table 2. As expected, most unknowns are open class words: proper names, nouns or adjectives.

Regarding unknown analyses, in our annotated corpus, we found 3% of the 100K token instances were missing the correct analysis in the lexicon (3.65% of the token types). The POS distribution of the unknown analyses is listed in Table 2. The high rate of unknown analyses for prepositions at about 3% is a specific phenomenon in Hebrew, where prepositions are often prefixes agglutinated to the first word of the noun phrase they head. We observe the very low rate of unknown verbs (2%) – which are well marked morphologically in Hebrew, and where the rate of neologism introduction seems quite low.

This evidence illustrates the need for resolution of unknowns: The naive policy of selecting ‘proper name’ for all unknowns will cover only half of the errors caused by unknown tokens, *i.e.*, 30% of the whole unknown tokens and analyses. The other 70% of the unknowns ( 5.3% of the words in the text in our experiments) will be assigned a wrong tag.

As a result of this observation, our strategy is to focus on full morphological analysis for unknown tokens and apply a proper name classifier for unknown analyses and unknown tokens. In this paper, we investigate various methods for achieving full morphological analysis distribution for unknown tokens. The methods are not based on an annotated corpus, nor on hand-crafted rules, but instead exploit the distribution of words in an available lexicon and the letter similarity of the unknown words with known words.

<sup>3</sup>Transcription according to Ornan (2002)

Category	Examples	Distribution	
		Types	Instances
Proper names	' <i>asulin</i> (family name) אסולין ' <i>a'udi</i> (Audi) אאודי	40%	48%
Neologisms	' <i>agabi</i> (incidental) אגבי <i>tizmur</i> (orchestration) תזמור	30%	32%
Abbreviation	<i>mz"p</i> (DIFS) מז"פ <i>kb"t</i> (security officer) קב"ט	2.4%	7.8%
Foreign	<i>presentacyah</i> (presentation) פרזנטציה ' <i>a'ut</i> (out) אאוט right	3.8%	5.8%
Wrong spelling	' <i>abibba'ahronah</i> (springatlast) אביבבאחרונה ' <i>idiqacyot</i> (idication) אידיקציות <i>ryušalaim</i> (Rejusalem) ריושלים	1.2%	4%
Alternative spelling	' <i>opyynim</i> (typical) אופיינים <i>priwwilegyah</i> (privilege ) פריווילגיה	3.5%	3%
Tokenization	<i>ha"sap</i> (the"threshold) ה"ספ ' <i>al/17</i> (on/17) על/71	8%	2%

Table 1: Unknown Hebrew token categories and distribution.

Part of Speech	Unknown Tokens	Unknown Analyses	Total
Proper name	31.8%	24.4%	56.2%
Noun	12.6%	1.6%	14.2%
Adjective	7.1%	1.7%	8.8%
Junk	3.0%	1.3%	4.3%
Numeral	1.1%	2.3%	3.4%
Preposition	0.3%	2.8%	3.1%
Verb	1.8%	0.4%	2.2%
Adverb	0.9%	0.9%	1.8%
Participle	0.4%	0.8%	1.2%
Copula	/	0.8%	0.8%
Quantifier	0.3%	0.4%	0.7%
Modal	0.3%	0.4%	0.7%
Conjunction	0.1%	0.5%	0.6%
Negation	/	0.6%	0.6%
Foreign	0.2%	0.4%	0.6%
Interrogative	0.1%	0.4%	0.5%
Prefix	0.3%	0.2%	0.5%
Pronoun	/	0.5%	0.5%
Total	60%	40%	100%

Table 2: Unknowns Hebrew POS Distribution.

## 2 Previous Work

Most of the work that dealt with unknowns in the last decade focused on unknown tokens (OOV). A naive approach would assign all possible analyses for each unknown token with uniform distribution, and continue disambiguation on the basis of a learned model with this initial distribution. The performance of a tagger with such a policy is actually poor: there are dozens of tags in the tagset (3,561 in the case of Hebrew full morphological disambiguation) and only a few of them may match a given token. Several heuristics were developed to reduce the possibility space and to assign a distribution for the remaining analyses.

Weischedel et al. (1993) combine several heuristics in order to estimate the token generation probability according to various types of information – such as the characteristics of particular tags with respect to unknown tokens (basically the distribution shown in Table 2), and simple spelling features: capitalization, presence of hyphens and specific suffixes. An accuracy of 85% in resolving unknown tokens was reported. Dermatas and Kokkinakis (1995) suggested a method for guessing unknown tokens based on the distribution of the hapax legomenon, and reported an accuracy of 66% for English. Mikheev (1997) suggested a guessing-rule technique, based on prefix morphological rules, suffix morphological rules, and ending-guessing rules. These rules are learned automatically from raw text. They reported a tagging accuracy of about 88%. Thede and Harper (1999) extended a second-order HMM model with a  $C = c_{k,i}$  matrix, in order to encode the probability of a token with a suffix  $s_k$  to be generated by a tag  $t_i$ . An accuracy of about 85% was reported.

Nakagawa (2004) combine word-level and character-level information for Chinese and Japanese word segmentation. At the word level, a segmented word is attached to a POS, where the character model is based on the observed characters and their classification: **B**egin of word, **I**n the middle of a word, **E**nd of word, the character is a word itself **S**. They apply Baum-Welch training over a segmented corpus, where the segmentation of each word and its character classification is observed, and the POS tagging is ambiguous. The segmentation

(of all words in a given sentence) and the POS tagging (of the known words) is based on a Viterbi search over a lattice composed of all possible word segmentations and the possible classifications of all observed characters. Their experimental results show that the method achieves high accuracy over state-of-the-art methods for Chinese and Japanese word segmentation. Hebrew also suffers from ambiguous segmentation of agglutinated tokens into significant words, but word formation rules seem to be quite different from Chinese and Japanese. We also could not rely on the existence of an annotated corpus of segmented word forms.

Habash and Rambow (2006) used the root+pattern+features representation of Arabic tokens for morphological analysis and generation of Arabic dialects, which have no lexicon. They report high recall (95%–98%) but low precision (37%–63%) for token types and token instances, against gold-standard morphological analysis. We also exploit the morphological patterns characteristic of semitic morphology, but extend the guessing of morphological features by using contextual features. We also propose a method that relies exclusively on learned character-level features and contextual features, and eventually reaches the same performance as the patterns-based approach.

Mansour et al. (2007) combine a lexicon-based tagger (such as MorphTagger (Bar-Haim et al., 2005)), and a character-based tagger (such as the data-driven ArabicSVM (Diab et al., 2004)), which includes character features as part of its classification model, in order to extend the set of analyses suggested by the analyzer. For a given sentence, the lexicon-based tagger is applied, selecting one tag for a token. In case the ranking of the tagged sentence is lower than a threshold, the character-based tagger is applied, in order to produce new possible analyses. They report a very slight improvement on Hebrew and Arabic supervised POS taggers.

Resolution of Hebrew unknown tokens, over a large number of tags in the tagset (3,561) requires a much richer model than the heuristics used for English (for example, the capitalization feature which is dominant in English does not exist in Hebrew). Unlike Nakagawa, our model does not use any segmented text, and, on the other hand, it aims to select full morphological analysis for each token,

including unknowns.

### 3 Method

Our objective is: given an unknown word, provide a distribution of possible tags that can serve as the analysis of the unknown word. This unknown analysis step is performed at training and testing time. We do not attempt to disambiguate the word – but only to provide a distribution of tags that will be disambiguated by the regular EM-HMM mechanism.

We examined three models to construct the distribution of tags for unknown words, that is, whenever the KC analyzer does not return any candidate analysis, we apply these models to produce possible tags for the token  $p(t|w)$ :

**Letters** A maximum entropy model is built for all unknown tokens in order to estimate their tag distribution. The model is trained on the known tokens that appear in the corpus. For each analysis of a known token, the following features are extracted: (1) unigram, bigram, and trigram letters of the base-word (for each analysis, the base-word is the token without prefixes), together with their index relative to the start and end of the word. For example, the n-gram features extracted for the word abc are { a:1 b:2 c:3 a:-3 b:-2 c:-1 ab:1 bc:2 ab:-2 bc:-1 abc:1 abc:-1 }; (2) the prefixes of the base-word (as a single feature); (3) the length of the base-word. The class assigned to this set of features, is the analysis of the base-word. The model is trained on all the known tokens of the corpus, each token is observed with its possible POS-tags once for each of its occurrences. When an unknown token is found, the model is applied as follows: all the possible linguistic prefixes are extracted from the token (one of the 76 prefix sequences that can occur in Hebrew); if more than one such prefix is found, the token is analyzed for each possible prefix. For each possible such segmentation, the full feature vector is constructed, and submitted to the Maximum Entropy model. We hypothesize a uniform distribution among the possible segmentations and aggregate a distribution of possible tags for the analysis. If the proposed tag of the base-word is never found in the corpus preceded by the identified prefix, we remove this possible analysis. The eventual outcome of the

model application is a set of possible full morphological analyses for the token – in exactly the same format as the morphological analyzer provides.

**Patterns** Word formation in Hebrew is based on root+pattern and affixation. Patterns can be used to identify the lexical category of unknowns, as well as other inflectional properties. Nir (1993) investigated word-formation in Modern Hebrew with a special focus on neologisms; the most common word-formation patterns he identified are summarized in Table 3. A naive approach for unknown resolution would add all analyses that fit any of these patterns, for any given unknown token. As recently shown by Habash and Rambow (2006), the precision of such a strategy can be pretty low. To address this lack of precision, we learn a maximum entropy model on the basis of the following binary features: one feature for each pattern listed in column **Formation** of Table 3 (40 distinct patterns) and one feature for “no pattern”.

**Pattern-Letters** This maximum entropy model is learned by combining the features of the letters model and the patterns model.

**Linear-Context-based  $p(t|c)$  approximation** The three models above are context free. The linear-context model exploits information about the lexical context of the unknown words: to estimate the probability for a tag  $t$  given a context  $c$  –  $p(t|c)$  – based on all the words in which a context occurs, the algorithm works on the known words in the corpus, by starting with an initial tag-word estimate  $p(t|w)$  (such as the morpho-lexical approximation, suggested by Levinger et al. (1995)), and iteratively re-estimating:

$$\begin{aligned}\hat{p}(t|c) &= \frac{\sum_{w \in W} p(t|w)p(w|c)}{Z} \\ \hat{p}(t|w) &= \frac{\sum_{c \in C} p(t|c)p(c|w)allow(t, w)}{Z}\end{aligned}$$

where  $Z$  is a normalization factor,  $W$  is the set of all words in the corpus,  $C$  is the set of contexts.  $allow(t, w)$  is a binary function indicating whether  $t$  is a valid tag for  $w$ .  $p(c|w)$  and  $p(w|c)$  are estimated via raw corpus counts.

Loosely speaking, the probability of a tag given a context is the average probability of a tag given any



Category	Formation		Example
Verb	Template	'iCCeC	'ibhen (diagnosed) אבחן
		miCCeC	mihzer (recycled) מחזר
		CiCCen	timren (manipulated) תמרן
		CiCCet	tiknet (programmed) תכנת
		tiCCeC	ti'arek (dated) תארך
Participle	Template	meCuCaca	mšwhzar (reconstructed) משוחזר
		muCCaC	muqlaṭ (recorded) מוקלט
		maCCiC	malbin (whitening) מלבין
Noun	Suffixation	ut	ḥaluciyut (pioneership) חלוציות
		ay	yomanay (duty officer) יומנאי
		an	'egropan (boxer) אגרופן
		on	paḥon (shack) פחון
		iya	marakiyah (soup tureen) מרקיה
		it	ṭiyulit (open touring vehicle) טיולית
		a	lomdah (courseware) לומדה
	Template	maCCeC	mašneq (choke) משנק
		maCCeCa	madgera (incubator) מדגרה
		miCCaC	mis'ap (branching) מסעף
		miCCaCa	mignana (defensive fighting) מגננה
		CeCeC <sup>a</sup>	peleṭ (output) פלט
		tiCCoCet	tiproset (distribution) תפרוסת
		taCCiC	tahriṭ (engraving) תחריט
		taCCuCa	tabru'ah (sanitation) תברואה
		miCCeCet	micrepet (leotard) מצרפת
		CCiC	crir (dissonance) צריר
		CaCCan	balšan (linguist) בלשן
		CaCeCet	šaḥemet (cirrhosis) שחמת
		CiCul	ṭibu' (ringing) טיבוע
		haCCaCa	hanpaša (animation) הנפשה
		heCCeC	het'em (agreement) התאם
		Adjective	Suffixation <sup>b</sup>
ani	yehidani (individual) יחידני		
oni	ṭelewizyoni <sup>c</sup> (televisional) טלוויזיוני		
a'i	yeḏida'i (unique) יחידאי		
ali	šṭudentiali (student) סטודנטיאלי		
Template	C <sub>1</sub> C <sub>2</sub> aC <sub>3</sub> C <sub>2</sub> aC <sub>3</sub> <sup>d</sup>		metaqtaq (sweetish) מתקתק
	CaCuC		rapus (flaccid) רפוס
Adverb	Suffixation	ot	qcarot (briefly) קצרות
		it	miyadit (immediately) מידית
	Prefixation	b	bekeip (with fun) בכיף

<sup>a</sup>CoCeC variation: עותק 'weq (a copy).

<sup>b</sup>The feminine form is made by the *t* and *iya* suffixes: יחידנית yehidanit (individual), נוצרייה nwcryia (Christian).

<sup>c</sup>In the feminine form, the last *h* of the original noun is omitted.

<sup>d</sup>C<sub>1</sub>C<sub>2</sub>aC<sub>3</sub>C<sub>2</sub>oC<sub>3</sub> variation: קטנטון qtanṭwn (tiny).

Table 3: Common Hebrew Neologism Formations.

Model	Analysis Set			Morphological Disambiguation
	Coverage	Ambiguity	Probability	
Baseline	50.8%	<b>1.5</b>	<b>0.48</b>	57.3%
Pattern	82.8%	20.4	0.10	66.8%
Letter	76.7%	5.9	0.32	69.1%
Pattern-Letter	84.1%	10.4	0.25	<b>69.8%</b>
WordContext-Pattern	84.4%	21.7	0.12	66.5%
TagContext-Pattern	85.3%	23.5	0.19	64.9%
WordContext-Letter	80.7%	7.94	0.30	<b>69.7%</b>
TagContext-Letter	83.1%	7.8	0.22	66.9%
WordContext-Pattern-Letter	85.2%	12.0	0.24	68.8%
TagContext-Pattern-Letter	<b>86.1%</b>	14.3	0.18	62.1%

Table 4: Evaluation of unknown token full morphological analysis.

of the words appearing in that context, and similarly the probability of a tag given a word is the averaged probability of that tag in all the (reliable) contexts in which the word appears. We use the function  $allow(t, w)$  to control the tags (ambiguity class) allowed for each word, as given by the lexicon.

For a given word  $w_i$  in a sentence, we examine two types of contexts: **word context**  $w_{i-1}, w_{i+1}$ , and **tag context**  $t_{i-1}, t_{i+1}$ . For the case of word context, the estimation of  $p(w|c)$  and  $p(c|w)$  is simply the relative frequency over all the events  $w1, w2, w3$  occurring at least 10 times in the corpus. Since the corpus is not tagged, the relative frequency of the tag contexts is not observed, instead, we use the context-free approximation of each word-tag, in order to determine the frequency weight of each tag context event. For example, given the sequence תגובה לעומתית למדי *tgubah l'umatit lmadai* (a quite oppositional response), and the analyses set produced by the context-free approximation: *tgubah* [NN 1.0] *l'umatit* [] *lmadai* [RB 0.8, P1-NN 0.2]. The frequency weight of the context {NN RB} is  $1 * 0.8 = 0.8$  and the frequency weight of the context {NN P1-NN} is  $1 * 0.2 = 0.2$ .

## 4 Evaluation

For testing, we manually tagged the text which is used in the Hebrew Treebank (consisting of about 90K tokens), according to our tagging guideline (?).

We measured the effectiveness of the three models with respect to the tags that were assigned to the unknown tokens in our test corpus (the 'correct tag'),

according to three parameters: (1) The coverage of the model, *i.e.*, we count cases where  $p(t|w)$  contains the correct tag with a probability larger than 0.01; (2) the ambiguity level of the model, *i.e.*, the average number of analyses suggested for each token; (3) the average probability of the 'correct tag', according to the predicted  $p(t|w)$ . In addition, for each experiment, we run the full morphology disambiguation system where unknowns are analyzed according to the model.

Our baseline proposes the most frequent tag (proper name) for all possible segmentations of the token, in a uniform distribution. We compare the following models: the 3 context free models (patterns, letters and the combined patterns and letters) and the same models combined with the word and tag context models. Note that the context models have low coverage (about 40% for the word context and 80% for the tag context models), and therefore, the context models cannot be used on their own. The highest coverage is obtained for the combined model (tag context, pattern, letter) at 86.1%.

We first show the results for full morphological disambiguation, over 3,561 distinct tags in Table 4. The highest coverage is obtained for the model combining the tag context, patterns and letters models. The tag context model is more effective because it covers 80% of the unknown words, whereas the word context model only covers 40%. As expected, our simple baseline has the highest precision, since the most frequent proper name tag covers over 50% of the unknown words. The eventual effectiveness of

Model	Analysis Set			POS Tagging
	Coverage	Ambiguity	Probability	
Baseline	52.9%	<b>1.5</b>	<b>0.52</b>	60.6%
Pattern	87.4%	8.7	0.19	76.0%
Letter	80%	4.0	0.39	77.6%
Pattern-Letter	86.7%	6.2	0.32	<b>78.5%</b>
WordContext-Pattern	88.7%	8.8	0.21	75.8%
TagContext-Pattern	<b>89.5%</b>	9.1	0.14	73.8%
WordContext-Letter	83.8%	4.5	0.37	<b>78.2%</b>
TagContext-Letter	87.1%	5.7	0.28	75.2%
WordContext-Pattern-Letter	87.8	6.5	0.32	77.5%
TagContext-Pattern-Letter	89.0%	7.2	0.25	74%

Table 5: Evaluation of unknown token POS tagging.

the method is measured by its impact on the eventual disambiguation of the unknown words. For full morphological disambiguation, our method achieves an error reduction of 30% (57% to 70%). Overall, with the level of 4.5% of unknown words observed in our corpus, the algorithm we have developed contributes to an error reduction of 5.5% for full morphological disambiguation.

The best result is obtained for the model combining pattern and letter features. However, the model combining the word context and letter features achieves almost identical results. This is an interesting result, as the pattern features encapsulate significant linguistic knowledge, which apparently can be approximated by a purely distributional approximation.

While the disambiguation level of 70% is lower than the rate of 85% achieved in English, it must be noted that the task of full morphological disambiguation in Hebrew is much harder – we manage to select one tag out of 3,561 for unknown words as opposed to one out of 46 in English. Table 5 shows the result of the disambiguation when we only take into account the POS tag of the unknown tokens. The same models reach the best results in this case as well (Pattern+Letters and WordContext+Letters). The best disambiguation result is 78.5% – still much lower than the 85% achieved in English. The main reason for this lower level is that the task in Hebrew includes segmentation of prefixes and suffixes in addition to POS classification. We are currently investigating models that will take into account the

specific nature of prefixes in Hebrew (which encode conjunctions, definite articles and prepositions) to better predict the segmentation of unknown words.

## 5 Conclusion

We have addressed the task of computing the distribution  $p(t|w)$  for unknown words for full morphological disambiguation in Hebrew. The algorithm we have proposed is language independent: it exploits a maximum entropy letters model trained over the known words observed in the corpus and the distribution of the unknown words in known tag contexts, through iterative approximation. The algorithm achieves 30% error reduction on disambiguation of unknown words over a competitive baseline (to a level of 70% accurate full disambiguation of unknown words). We have also verified that taking advantage of a strong language-specific model of morphological patterns provides the same level of disambiguation. The algorithm we have developed exploits distributional information latent in a wide-coverage lexicon and large quantities of unlabeled data.

We observe that the task of analyzing unknown tokens for POS in Hebrew remains challenging when compared with English (78% vs. 85%). We hypothesize this is due to the highly ambiguous pattern of prefixation that occurs widely in Hebrew and are currently investigating syntagmatic models that exploit the specific nature of agglutinated prefixes in Hebrew.

## References

- Meni Adler. 2007. *Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach*. Ph.D. thesis, Ben-Gurion University of the Negev, Beer-Sheva, Israel.
- Roy Bar-Haim, Khalil Sima'an, and Yoad Winter. 2005. Choosing an optimal architecture for segmentation and pos-tagging of modern Hebrew. In *Proceedings of ACL-05 Workshop on Computational Approaches to Semitic Languages*.
- Tim Buckwalter. 2004. Buckwalter Arabic morphological analyzer, version 2.0.
- Evangelos Dermatas and George Kokkinakis. 1995. Automatic stochastic tagging of natural language texts. *Computational Linguistics*, 21(2):137–163.
- Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of Arabic text: From raw text to base phrase chunks. In *Proceeding of HLT-NAACL-04*.
- Michael Elhadad, Yael Netzer, David Gabay, and Meni Adler. 2005. Hebrew morphological tagging guidelines. Technical report, Ben-Gurion University, Dept. of Computer Science.
- Nizar Habash and Owen Rambow. 2006. Magead: A morphological analyzer and generator for the arabic dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 681–688, Sydney, Australia, July. Association for Computational Linguistics.
- Moshe Levinger, Uzi Ornan, and Alon Itai. 1995. Learning morpholexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics*, 21:383–404.
- Saib Mansour, Khalil Sima'an, and Yoad Winter. 2007. Smoothing a lexicon-based pos tagger for Arabic and Hebrew. In *ACL07 Workshop on Computational Approaches to Semitic Languages*, Prague, Czech Republic.
- Andrei Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Computational Linguistics*, 23(3):405–423.
- Tetsuji Nakagawa. 2004. Chinese and Japanese word segmentation using word-level and character-level information. In *Proceedings of the 20th international conference on Computational Linguistics*, Geneva.
- Raphael Nir. 1993. *Word-Formation in Modern Hebrew*. The Open University of Israel, Tel-Aviv, Israel.
- Uzi Ornan. 2002. Hebrew in Latin script. *Lěšoněnu*, LXIV:137–151. (in Hebrew).
- Scott M. Thede and Mary P. Harper. 1999. A second-order hidden Markov model for part-of-speech tagging. In *Proceeding of ACL-99*.
- R. Weischedel, R. Schwartz, J. Palmucci, M. Meteer, and L. Ramshaw. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19:359–382.

# Unsupervised Multilingual Learning for Morphological Segmentation

Benjamin Snyder and Regina Barzilay

Computer Science and Artificial Intelligence Laboratory

Massachusetts Institute of Technology

{bsnyder, regina}@csail.mit.edu

## Abstract

For centuries, the deep connection between languages has brought about major discoveries about human communication. In this paper we investigate how this powerful source of information can be exploited for unsupervised language learning. In particular, we study the task of morphological segmentation of multiple languages. We present a non-parametric Bayesian model that jointly induces morpheme segmentations of each language under consideration and at the same time identifies cross-lingual morpheme patterns, or *abstract morphemes*. We apply our model to three Semitic languages: Arabic, Hebrew, Aramaic, as well as to English. Our results demonstrate that learning morphological models in tandem reduces error by up to 24% relative to monolingual models. Furthermore, we provide evidence that our joint model achieves better performance when applied to languages from the same family.

## 1 Introduction

For centuries, the deep connection between human languages has fascinated linguists, anthropologists and historians (Eco, 1995). The study of this connection has made possible major discoveries about human communication: it has revealed the evolution of languages, facilitated the reconstruction of proto-languages, and led to understanding language universals.

The connection between languages should be a powerful source of information for automatic linguistic analysis as well. In this paper we investigate two questions: (i) Can we exploit cross-lingual correspondences to improve unsupervised language

learning? (ii) Will this joint analysis provide more or less benefit when the languages belong to the same family?

We study these two questions in the context of unsupervised morphological segmentation, the automatic division of a word into morphemes (the basic units of meaning). For example, the English word *misunderstanding* would be segmented into *mis - understand - ing*. This task is an informative testbed for our exploration, as strong correspondences at the morphological level across various languages have been well-documented (Campbell, 2004).

The model presented in this paper automatically induces a segmentation and morpheme alignment from a multilingual corpus of short parallel phrases.<sup>1</sup> For example, given parallel phrases meaning *in my land* in English, Arabic, Hebrew, and Aramaic, we wish to segment and align morphemes as follows:

English:	<i>in my land</i>
Arabic:	<i>fy ard - y</i>
Hebrew:	<i>b - arš - y</i>
Aramaic:	<i>b - ar<sup>c</sup> - y</i>

This example illustrates the potential benefits of unsupervised multilingual learning. The three Semitic languages use cognates (words derived from a common ancestor) to represent the word *land*. They also use an identical suffix (-y) to represent the first person possessive pronoun (*my*). These similarities in form should guide the model by constraining

<sup>1</sup>In this paper, we focus on bilingual models. The model can be extended to handle several languages simultaneously as in this example.

the space of joint segmentations. The corresponding English phrase lacks this resemblance to its Semitic counterparts. However, in this as in many cases, no segmentation is required for English as all the morphemes are expressed as individual words. For this reason, English should provide a strong source of disambiguation for highly inflected languages, such as Arabic and Hebrew.

In general, we pose the following question. In which scenario will multilingual learning be most effective? Will it be for related languages, which share a common core of linguistic features, or for distant languages, whose linguistic divergence can provide strong sources of disambiguation?

As a first step towards answering this question, we propose a model which can take advantage of both similarities and differences across languages. This joint bilingual model identifies optimal morphemes for two languages and at the same time finds compact multilingual representations. For each language in the pair, the model favors segmentations which yield high frequency morphemes. Moreover, bilingual morpheme pairs which consistently share a common semantic or syntactic function are treated as *abstract morphemes*, generated by a single language-independent process. These abstract morphemes are induced automatically by the model from recurring bilingual patterns. For example, in the case above, the tuple  $(in, fy, b-, b-)$  would constitute one of three abstract morphemes in the phrase. When a morpheme occurs in one language without a direct counterpart in the other language, our model can explain away the stray morpheme as arising through a language-specific process.

To achieve this effect in a probabilistic framework, we formulate a hierarchical Bayesian model with Dirichlet Process priors. This framework allows us to define priors over the infinite set of possible morphemes in each language. In addition, we define a prior over abstract morphemes. This prior can incorporate knowledge of the phonetic relationship between the two alphabets, giving potential cognates greater prior likelihood. The resulting posterior distributions concentrate their probability mass on a small group of recurring and stable patterns within and between languages.

We test our model on a multilingual corpus of short parallel phrases drawn from the Hebrew Bible

and Arabic, Aramaic, and English translations. The Semitic language family, of which Hebrew, Arabic, and Aramaic are members, is known for a highly productive morphology (Bravmann, 1977). Our results indicate that cross-lingual patterns can indeed be exploited successfully for the task of unsupervised morphological segmentation. When modeled in tandem, gains are observed for all language pairs, reducing relative error by as much as 24%. Furthermore, our experiments show that both related and unrelated language pairs benefit from multilingual learning. However, when common structures such as phonetic correspondences are explicitly modeled, related languages provide the most benefit.

## 2 Related Work

**Multilingual Language Learning** Recently, the availability of parallel corpora has spurred research on multilingual analysis for a variety of tasks ranging from morphology to semantic role labeling (Yarowsky et al., 2000; Diab and Resnik, 2002; Xi and Hwa, 2005; Padó and Lapata, 2006). Most of this research assumes that one language has annotations for the task of interest. Given a parallel corpus, the annotations are projected from this source language to its counterpart, and the resulting annotations are used for supervised training in the target language. In fact, Rogati et al., (2003) employ this method to learn arabic morphology assuming annotations provided by an English stemmer.

An alternative approach has been proposed by Feldman, Hana and Brew (2004; 2006). While their approach does not require a parallel corpus it does assume the availability of annotations in one language. Rather than being fully projected, the source annotations provide co-occurrence statistics used by a model in the resource-poor target language. The key assumption here is that certain distributional properties are invariant across languages from the same language families. An example of such a property is the distribution of part-of-speech bigrams. Hana et al., (2004) demonstrate that adding such statistics from an annotated Czech corpus improves the performance of a Russian part-of-speech tagger over a fully unsupervised version.

The approach presented here differs from previous work in two significant ways. First, we do

not assume supervised data in any of the languages. Second, we learn a single multilingual model, rather than asymmetrically handling one language at a time. This design allows us to capitalize on structural regularities across languages for the mutual benefit of each language.

### Unsupervised Morphological Segmentation

Unsupervised morphology is an active area of research (Schone and Jurafsky, 2000; Goldsmith, 2001; Adler and Elhadad, 2006; Creutz and Lagus, 2007; Dasgupta and Ng, 2007).

Most existing algorithms derive morpheme lexicons by identifying recurring patterns in string distribution. The goal is to optimize the compactness of the data representation by finding a small lexicon of highly frequent strings. Our work builds on probabilistic segmentation approaches such as Morfessor (Creutz and Lagus, 2007). In these approaches, models with short description length are preferred. Probabilities are computed for both the morpheme lexicon and the representation of the corpus conditioned on the lexicon. A locally optimal segmentation is identified using a task-specific greedy search.

In contrast to previous approaches, our model induces morphological segmentation for multiple related languages simultaneously. By representing morphemes abstractly through the simultaneous alignment and segmentation of data in two languages, our algorithm capitalizes on deep connections between morpheme usage across different languages.

### 3 Multilingual Morphological Segmentation

The underlying assumption of our work is that structural commonality across different languages is a powerful source of information for morphological analysis. In this section, we provide several examples that motivate this assumption.

The main benefit of joint multilingual analysis is that morphological structure ambiguous in one language is sometimes explicitly marked in another language. For example, in Hebrew, the preposition meaning “in”, *b-*, is always prefixed to its nominal argument. On the other hand, in Arabic, the most common corresponding particle is *fy*, which appears as a separate word. By modeling cross-

lingual morpheme alignments while simultaneously segmenting, the model effectively propagates information between languages and in this case would be encouraged to segment the Hebrew prefix *b-*.

Cognates are another important means of disambiguation in the multilingual setting. Consider translations of the phrase “...and they wrote it...”:

- Hebrew: *w-ktb-w ath*
- Arabic: *f-ktb-w-ha*

In both languages, the trilateral root *ktb* is used to express the act of writing. By considering the two phrases simultaneously, the model can be encouraged to split off the respective Hebrew and Arabic prefixes *w-* and *f-* in order to properly align the cognate root *ktb*.

In the following section, we describe a model that can model both generic cross-lingual patterns (*fy* and *b-*), as well as cognates between related languages (*ktb* for Hebrew and Arabic).

## 4 Model

**Overview** In order to simultaneously model probabilistic dependencies across languages as well as morpheme distributions within each language, we employ a hierarchical Bayesian model.<sup>2</sup>

Our segmentation model is based on the notion that stable recurring string patterns within words are indicative of morphemes. In addition to learning independent morpheme patterns for each language, the model will prefer, when possible, to join together frequently occurring bilingual morpheme pairs into single *abstract morphemes*. The model is fully unsupervised and is driven by a preference for stable and high frequency cross-lingual morpheme patterns. In addition the model can incorporate character-to-character phonetic correspondences between alphabets as prior information, thus allowing the implicit modeling of cognates.

Our aim is to induce a model which concentrates probability on highly frequent patterns while still allowing for the possibility of those previously unseen. Dirichlet processes are particularly suitable for such conditions. In this framework, we can encode

<sup>2</sup>In (Snyder and Barzilay, 2008) we consider the use of this model in the case where supervised data in one or more languages is available.

prior knowledge over the infinite sets of possible morpheme strings as well as abstract morphemes. Distributions drawn from a Dirichlet process nevertheless produce sparse representations with most probability mass concentrated on a small number of observed and predicted patterns. Our model utilizes a Dirichlet process prior for each language, as well as for the cross-lingual links (*abstract morphemes*). Thus, a distribution over morphemes and morpheme alignments is first drawn from the set of Dirichlet processes and then produces the observed data. In practice, we never deal with such distributions directly, but rather integrate over them during Gibbs sampling.

In the next section we describe our model’s “generative story” for producing the data we observe. We formalize our model in the context of two languages  $\mathcal{E}$  and  $\mathcal{F}$ . However, the formulation can be extended to accommodate evidence from multiple languages as well. We provide an example of parallel phrase generation in Figure 1.

**High-level Generative Story** We have a parallel corpus of several thousand short phrases in the two languages  $\mathcal{E}$  and  $\mathcal{F}$ . Our model provides a generative story explaining how these parallel phrases were probabilistically created. The core of the model consists of three components: a distribution  $A$  over bilingual morpheme pairs (*abstract morphemes*), a distribution  $E$  over stray morphemes in language  $\mathcal{E}$  occurring without a counterpart in language  $\mathcal{F}$ , and a similar distribution  $F$  for stray morphemes in language  $\mathcal{F}$ .

As usual for hierarchical Bayesian models, the generative story begins by drawing the model parameters themselves – in our case the three distributions  $A$ ,  $E$ , and  $F$ . These three distributions are drawn from three separate Dirichlet processes, each with appropriately defined base distributions. The Dirichlet processes ensure that the resulting distributions concentrate their probability mass on a small number of morphemes while holding out reasonable probability for unseen possibilities.

Once  $A$ ,  $E$ , and  $F$  have been drawn, we model our parallel corpus of short phrases as a series of independent draws from a phrase-pair generation model. For each new phrase-pair, the model first chooses the number and type of morphemes to be

generated. In particular, it must choose how many unaligned stray morphemes from language  $\mathcal{E}$ , unaligned stray morphemes from language  $\mathcal{F}$ , and abstract morphemes are to compose the parallel phrases. These three numbers, respectively denoted as  $m$ ,  $n$ , and  $k$ , are drawn from a Poisson distribution. This step is illustrated in Figure 1 part (a).

The model then proceeds to independently draw  $m$  language  $\mathcal{E}$  morphemes from distribution  $E$ ,  $n$  language- $\mathcal{F}$  morphemes from distribution  $F$ , and  $k$  abstract morphemes from distribution  $A$ . This step is illustrated in part (b) of Figure 1.

The  $m + k$  resulting language- $\mathcal{E}$  morphemes are then ordered and fused to form a phrase in language  $\mathcal{E}$ , and likewise for the  $n + k$  resulting language- $\mathcal{F}$  morphemes. The ordering and fusing decisions are modeled as draws from a uniform distribution over the set of all possible orderings and fusings for sizes  $m$ ,  $n$ , and  $k$ . These final steps are illustrated in parts (c)-(d) of Figure 1. Now we describe the model more formally.

**Stray Morpheme Distributions** Sometimes a morpheme occurs in a phrase in one language without a corresponding foreign language morpheme in the parallel phrase. We call these “stray morphemes,” and we employ language-specific morpheme distributions to model their generation.

For each language, we draw a distribution over all possible morphemes (finite-length strings composed of characters in the appropriate alphabet) from a Dirichlet process with concentration parameter  $\alpha$  and base distribution  $P_e$  or  $P_f$  respectively:

$$\begin{aligned} E|\alpha, P_e &\sim DP(\alpha, P_e) \\ F|\alpha, P_f &\sim DP(\alpha, P_f) \end{aligned}$$

The base distributions  $P_e$  and  $P_f$  can encode prior knowledge about the properties of morphemes in each of the two languages, such as length and character n-grams. For simplicity, we use a geometric distribution over the length of the string with a final end-morpheme character. The distributions  $E$  and  $F$  which result from the respective Dirichlet processes place most of their probability mass on a small number of morphemes with the degree of concentration



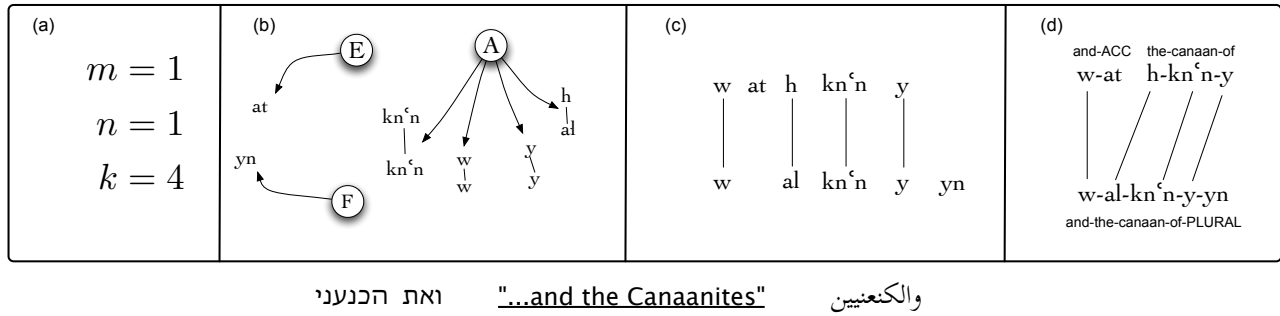


Figure 1: Generation process for a parallel bilingual phrase, with Hebrew shown on top and Arabic on bottom. (a) First the numbers of stray ( $m$  and  $n$ ) and abstract ( $k$ ) morphemes are drawn from a Poisson distribution. (b) Stray morphemes are then drawn from  $E$  and  $F$  (language-specific distributions) and abstract morphemes are drawn from  $A$ . (c) The resulting morphemes are ordered. (d) Finally, some of the contiguous morphemes are fused into words.

controlled by the prior  $\alpha$ . Nevertheless, some non-zero probability is reserved for every possible string.

We note that these single-language morpheme distributions also serve as monolingual segmentation models, and similar models have been successfully applied to the task of word boundary detection (Goldwater et al., 2006).

**Abstract Morpheme Distribution** To model the connections between morphemes across languages, we further define a model for bilingual morpheme pairs, or *abstract morphemes*. This model assigns probabilities to all pairs of morphemes – that is, all pairs of finite strings from the respective alphabets –  $(e, f)$ . Intuitively, we wish to assign high probability to pairs of morphemes that play similar syntactic or semantic roles (e.g.  $(fy, b-)$  for “in” in Arabic and Hebrew). These morpheme pairs can thus be viewed as representing *abstract morphemes*. As with the stray morpheme models, we wish to define a distribution which concentrates probability mass on a small number of highly co-occurring morpheme pairs while still holding out some probability for all other pairs.

We define this abstract morpheme model  $A$  as a draw from another Dirichlet process:

$$\begin{aligned}
 A | \alpha', P' &\sim DP(\alpha', P') \\
 (e, f) &\sim A
 \end{aligned}$$

As before, the resulting distribution  $A$  will give non-zero probability to all abstract morphemes

$(e, f)$ . The base distribution  $P'$  acts as a prior on such pairs. To define  $P'$ , we can simply use a mixture of geometric distributions in the lengths of the component morphemes. However, if the languages  $\mathcal{E}$  and  $\mathcal{F}$  are related and the regular phonetic correspondences between the letter in the two alphabets are known, then we can use  $P'$  to assign higher likelihood to potential cognates. In particular we define the prior  $P'(e, f)$  to be the probabilistic string-edit distance (Ristad and Yianilos, 1998) between  $e$  and  $f$ , using the known phonetic correspondences to parameterize the string-edit model. In particular, insertion and deletion probabilities are held constant for all characters, and substitution probabilities are determined based on the known sound correspondences.

We report results for both the simple geometric prior as well as the string-edit prior.

**Phrase Generation** To generate a bilingual parallel phrase, we first draw  $m$ ,  $n$ , and  $k$  independently from a Poisson distribution. These three integers represent the number and type of the morphemes that compose the parallel phrase, giving the number of stray morphemes in each language  $\mathcal{E}$  and  $\mathcal{F}$  and the number of coupled bilingual morpheme pairs, respectively.

$$m, n, k \sim \text{Poisson}(\lambda)$$

Given these values, we now draw the appropriate number of stray and abstract morphemes from the corresponding distributions:

$$\begin{aligned}
e_1, \dots, e_m &\sim E \\
f_1, \dots, f_n &\sim F \\
(e'_1, f'_1), \dots, (e'_k, f'_k) &\sim A
\end{aligned}$$

The sets of morphemes drawn for each language are then ordered:

$$\begin{aligned}
\tilde{e}_1, \dots, \tilde{e}_{m+k} &\sim ORDER|e_1, \dots, e_m, e'_1, \dots, e'_k \\
\tilde{f}_1, \dots, \tilde{f}_{n+k} &\sim ORDER|f_1, \dots, f_n, f'_1, \dots, f'_k
\end{aligned}$$

Finally the ordered morphemes are fused into the words that form the parallel phrases:

$$\begin{aligned}
w_1, \dots, w_s &\sim FUSE|\tilde{e}_1, \dots, \tilde{e}_{m+k} \\
v_1, \dots, v_t &\sim FUSE|\tilde{f}_1, \dots, \tilde{f}_{n+k}
\end{aligned}$$

To keep the model as simple as possible, we employ uniform distributions over the sets of orderings and fusings. In other words, given a set of  $r$  morphemes (for each language), we define the distribution over permutations of the morphemes to simply be  $ORDER(\cdot|r) = \frac{1}{r!}$ . Then, given a fixed morpheme order, we consider fusing each adjacent morpheme into a single word. Again, we simply model the distribution over the  $r - 1$  fusing decisions uniformly as  $FUSE(\cdot|r) = \frac{1}{2^{r-1}}$ .

**Implicit Alignments** Note that nowhere do we explicitly assign probabilities to morpheme alignments between parallel phrases. However, our model allows morphemes to be generated in precisely one of two ways: as a lone stray morpheme or as part of a bilingual abstract morpheme pair. Thus, our model implicitly assumes that each morpheme is either unaligned, or aligned to exactly one morpheme in the opposing language.

If we are given a parallel phrase with already segmented morphemes we can easily induce the distribution over alignments implied by our model. As we will describe in the next section, drawing from these induced alignment distributions plays a crucial role in our inference procedure.

**Inference** Given our corpus of short parallel bilingual phrases, we wish to make segmentation decisions which yield a set of morphemes with high joint probability. To assess the probability of a potential morpheme set, we need to marginalize over all possible alignments (i.e. possible abstract morpheme pairings and stray morpheme assignments). We also need to marginalize over all possible draws of the distributions  $A$ ,  $E$ , and  $F$  from their respective Dirichlet process priors. We achieve these aims by performing Gibbs sampling.

**Sampling** We follow (Neal, 1998) in the derivation of our blocked and collapsed Gibbs sampler. Gibbs sampling starts by initializing all random variables to arbitrary starting values. At each iteration, the sampler selects a random variable  $X_i$ , and draws a new value for  $X_i$  from the conditional distribution of  $X_i$  given the current value of the other variables:  $P(X_i|X_{-i})$ . The stationary distribution of variables derived through this procedure is guaranteed to converge to the true joint distribution of the random variables. However, if some variables can be jointly sampled, then it may be beneficial to perform block sampling of these variables to speed convergence. In addition, if a random variable is not of direct interest, we can avoid sampling it directly by marginalizing it out, yielding a collapsed sampler. We utilize variable blocking by jointly sampling multiple segmentation and alignment decisions. We also collapse our Gibbs sampler in the standard way, by using predictive posteriors marginalized over all possible draws from the Dirichlet processes (resulting in Chinese Restaurant Processes).

**Resampling** For each bilingual phrase, we resample each word in the phrase in turn. For word  $w$  in language  $\mathcal{E}$ , we consider at once all possible segmentations, and for each segmentation all possible alignments. We keep fixed the previously sampled segmentation decisions for all other words in the phrase as well as sampled alignments involving morphemes in other words. We are thus considering at once: all possible segmentations of  $w$  along with all possible alignments involving morphemes in  $w$  with some subset of previously sampled language- $\mathcal{F}$  morphemes.<sup>3</sup>

<sup>3</sup>We retain morpheme identities during resampling of the morpheme alignments. This procedure is technically just-

	Arabic			Hebrew		
	precision	recall	F-score	precision	recall	F-score
RANDOM	18.28	19.24	18.75	24.95	24.66	24.80
MORFESSOR	71.10	60.51	65.38	65.38	57.69	61.29
MONOLINGUAL	52.95	78.46	63.22	55.76	64.44	59.78
+ ARABIC/HEBREW	60.40	78.64	68.32	59.08	66.50	62.57
+ ARAMAIC	61.33	77.83	68.60	54.63	65.68	59.64
+ ENGLISH	63.19	74.79	68.49	60.20	64.42	62.23
+ ARAMAIC+PH	66.74	75.46	70.83	60.87	59.73	60.29
+ ARABIC/HEBREW+PH	67.75	77.29	<b>72.20</b>	64.90	62.87	<b>63.87</b>

Table 1: Precision, recall and F-score evaluated on Arabic and Hebrew. The first three rows provide baselines (random selection, an alternative state-of-the-art system, and the monolingual version of our model). The next three rows show the result of our bilingual model when one of Arabic, Hebrew, Aramaic, or English is added. The final two rows show the result of the bilingual model when character-to-character phonetic correspondences are used in the abstract morpheme prior.

The sampling formulas are easily derived as products of the relevant Chinese Restaurant Processes (with a minor adjustment to take into account the number of stray and abstract morphemes resulting from each decision). See (Neal, 1998) for general formulas for Gibbs sampling from distributions with Dirichlet process priors. All results reported are averaged over five runs using simulated annealing.

## 5 Experimental Set-Up

**Morpheme Definition** For the purpose of these experiments, we define *morphemes* to include conjunctions, prepositional and pronominal affixes, plural and dual suffixes, particles, definite articles, and roots. We do not model cases of infix morpheme transformations, as those cannot be modeled by linear segmentation.

**Dataset** As a source of parallel data, we use the Hebrew Bible and translations. For the Hebrew version, we use an edition distributed by Westminster Hebrew Institute (Groves and Lowery, 2006). This Bible edition is augmented by gold standard morphological analysis (including segmentation) performed by biblical scholars.

For the Arabic, Aramaic, and English versions,

we use the Van Dyke Arabic translation,<sup>4</sup> Targum Onkelos,<sup>5</sup> and the Revised Standard Version (Nelson, 1952), respectively. We obtained gold standard segmentations of the Arabic translation with a hand-crafted Arabic morphological analyzer which utilizes manually constructed word lists and compatibility rules and is further trained on a large corpus of hand-annotated Arabic data (Habash and Rambow, 2005). The accuracy of this analyzer is reported to be 94% for full morphological analyses, and 98%-99% when part-of-speech tag accuracy is not included. We don't have gold standard segmentations for the English and Aramaic portions of the data, and thus restrict our evaluation to Hebrew and Arabic.

To obtain our corpus of short parallel phrases, we preprocessed each language pair using the Giza++ alignment toolkit.<sup>6</sup> Given word alignments for each language pair, we extract a list of phrase pairs that form independent sets in the bipartite alignment graph. This process allows us to group together phrases like *fy şbah* in Arabic and *bbqr* in Hebrew while being reasonably certain that all the relevant morphemes are contained in the short extracted phrases. The number of words in such phrases ranges from one to four words in the Semitic languages and up to six words in English. Before performing any experiments, a manual inspection of

<sup>4</sup><http://www.arabicbible.com/bible/vandyke.htm>

<sup>5</sup><http://www.mechon-mamre.org/i/t/u/u0.htm>

<sup>6</sup><http://www.fjoch.com/GIZA++.html>

the generated parallel phrases revealed that many infrequent phrase pairs occurred merely as a result of noisy translation and alignment. Therefore, we eliminated all parallel phrases that occur fewer than five times. As a result of this process, we obtain 6,139 parallel short phrases in Arabic, Hebrew, Aramaic, and English. The average number of morphemes per word in the Hebrew data is 1.8 and is 1.7 in Arabic.

For the bilingual models which employs probabilistic string-edit distance as a prior on abstract morphemes, we parameterize the string-edit model with the chart of Semitic consonant relationships listed on page xxiv of (Thackston, 1999). All pairs of corresponding letters are given equal substitution probability, while all other letter pairs are given substitution probability of zero.

**Evaluation Methods** Following previous work, we evaluate the performance of our automatic segmentation algorithm using F-score. This measure is the harmonic mean of recall and precision, which are calculated on the basis of all possible segmentation points. The evaluation is performed on a random set of 1/5 of the parallel phrases which is unseen during the training phase. During testing, *we do not allow the models to consider any multilingual evidence*. This restriction allows us to simulate future performance on purely monolingual data.

**Baselines** Our primary purpose is to compare the performance of our bilingual model with its fully monolingual counterpart. However, to demonstrate the competitiveness of this baseline model, we also provide results using MORFESSOR (Creutz and Lagus, 2007), a state-of-the-art unsupervised system for morphological segmentation. While developed originally for Finnish, this system has been successfully applied to a range of languages including German, Turkish and English. The probabilistic formulation of this model is close to our monolingual segmentation model, but it uses a greedy search specifically designed for the segmentation task. We use the publicly available implementation of this system. To provide some idea of the inherent difficulty of this segmentation task, we also provide results from a random baseline which makes segmentation decisions based on a coin weighted with the true segmentation frequency.

## 6 Results

Table 1 shows the performance of the various automatic segmentation methods. The first three rows provide baselines, as mentioned in the previous section. Our primary baseline is MONOLINGUAL, which is the monolingual counterpart to our model and only uses the language-specific distributions  $E$  or  $F$ . The next three rows shows the performance of various bilingual models that don't use character-to-character phonetic correspondences to capture cognate information. We find that with the exception of the HEBREW(+ARAMAIC) pair, the bilingual models show marked improvement over MONOLINGUAL. We notice that in general, adding English – which has comparatively little morphological ambiguity – is about as useful as adding a more closely related Semitic language. However, once character-to-character phonetic correspondences are added as an abstract morpheme prior (final two rows), we find the performance of related language pairs outstrips English, reducing relative error over MONOLINGUAL by 10% and 24% for the Hebrew/Arabic pair.

## 7 Conclusions and Future Work

We started out by posing two questions: (i) Can we exploit cross-lingual patterns to improve unsupervised analysis? (ii) Will this joint analysis provide more or less benefit when the languages belong to the same family? The model and results presented in this paper answer the first question in the affirmative, at least for the task of morphological segmentation.

We also provided some evidence that considering closely related languages may be more beneficial than distant pairs *if* the model is able to explicitly represent shared language structure (the character-to-character phonetic correspondences in our case). In the future, we hope to apply similar multilingual models to other core unsupervised analysis tasks, including part-of-speech tagging and grammar induction, and to further investigate the role that language relatedness plays in such models.<sup>7</sup>

<sup>7</sup>We acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0415865) and the Microsoft Research Faculty Fellowship. Thanks to members of the MIT NLP group for enlightening discussion.

## References

- Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of the ACL/CONLL*, pages 665–672.
- M. M. Bravmann. 1977. *Studies in Semitic Philology*. Leiden:Brill.
- Lyle Campbell. 2004. *Historical Linguistics: An Introduction*. Cambridge: MIT Press.
- Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1).
- Sajib Dasgupta and Vincent Ng. 2007. Unsupervised part-of-speech acquisition for resource-scarce languages. In *Proceedings of the EMNLP-CoNLL*, pages 218–227.
- Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the ACL*, pages 255–262.
- Umberto Eco. 1995. *The Search for the Perfect Language*. Wiley-Blackwell.
- Anna Feldman, Jirka Hana, and Chris Brew. 2006. A cross-language approach to rapid creation of new morpho-syntactically annotated resources. In *Proceedings of LREC*.
- John A. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.
- Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the ACL*, pages 673–680.
- Alan Groves and Kirk Lowery, editors. 2006. *The Westminster Hebrew Bible Morphology Database*. Westminster Hebrew Institute, Philadelphia, PA, USA.
- Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the ACL*, pages 573–580.
- Jiri Hana, Anna Feldman, and Chris Brew. 2004. A resource-light approach to russian morphology: Tagging russian using czech resources. In *Proceedings of EMNLP*, pages 222–229.
- Radford M. Neal. 1998. Markov chain sampling methods for dirichlet process mixture models. Technical Report 9815, Dept. of Statistics and Dept. of Computer Science, University of Toronto, September.
- Thomas Nelson, editor. 1952. *The Holy Bible Revised Standard Version*. Thomas Nelson & Sons.
- Sebastian Padó and Mirella Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of ACL*, pages 1161 – 1168.
- Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532.
- Monica Rogati, J. Scott McCarley, and Yiming Yang. 2003. Unsupervised learning of arabic stemming using a parallel corpus. In *Proceedings of the ACL*, pages 391–398.
- Patrick Schone and Daniel Jurafsky. 2000. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the CoNLL*, pages 67–72.
- Benjamin Snyder and Regina Barzilay. 2008. Cross-lingual propagation for morphological analysis. In *Proceedings of AAAI*.
- Wheeler M. Thackston. 1999. *Introduction to Syriac*. Ibx Publishers.
- Chenhai Xi and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-english languages. In *Proceedings of HLT/EMNLP*, pages 851 – 858.
- David Yarowsky, Grace Ngai, and Richard Wicentowski. 2000. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT*, pages 161–168.

# EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start)\*

Yoav Goldberg and Meni Adler and Michael Elhadad

Ben Gurion University of the Negev

Department of Computer Science

POB 653 Be'er Sheva, 84105, Israel

{yoavg, adlerm, elhadad}@cs.bgu.ac.il

## Abstract

We address the task of unsupervised POS tagging. We demonstrate that good results can be obtained using the robust EM-HMM learner when provided with good initial conditions, even with incomplete dictionaries. We present a family of algorithms to compute effective initial estimations  $p(t|w)$ . We test the method on the task of full morphological disambiguation in Hebrew achieving an error reduction of 25% over a strong uniform distribution baseline. We also test the same method on the standard WSJ unsupervised POS tagging task and obtain results competitive with recent state-of-the-art methods, while using simple and efficient learning methods.

## 1 Introduction

The task of unsupervised (or semi-supervised) part-of-speech (POS) tagging is the following: given a dictionary mapping words in a language to their possible POS, and large quantities of unlabeled text data, learn to predict the correct part of speech for a given word in context. The only supervision given to the learning process is the dictionary, which in a realistic scenario, contains only part of the word types observed in the corpus to be tagged.

Unsupervised POS tagging has been traditionally approached with relative success (Merialdo, 1994; Kupiec, 1992) by HMM-based generative models, employing EM parameters estimation using the Baum-Welch algorithm. However, as recently noted

by Banko and Moore (2004), these works made use of filtered dictionaries: dictionaries in which only relatively probable analyses of a given word are preserved. This kind of filtering requires serious supervision: in theory, an expert is needed to go over the dictionary elements and filter out unlikely analyses. In practice, counts from an annotated corpus have been traditionally used to perform the filtering. Furthermore, these methods require rather comprehensive dictionaries in order to perform well.

In recent work, researchers try to address these deficiencies by using dictionaries with unfiltered POS-tags, and testing the methods on “diluted dictionaries” – in which many of the lexical entries are missing (Smith and Eisner, 2005) (SE), (Goldwater and Griffiths, 2007) (GG), (Toutanova and Johnson, 2008) (TJ).

All the work mentioned above focuses on unsupervised **English** POS tagging. The dictionaries are all derived from tagged English corpora (all recent work uses the WSJ corpus). As such, the setting of the research is artificial: there is no reason to perform unsupervised learning when an annotated corpus is available. The problem is rather approached as a workbench for exploring new learning methods. The result is a series of creative algorithms, that have steadily improved results on the same dataset: unsupervised CRF training using contrastive estimation (SE), a fully-bayesian HMM model that jointly performs clustering and sequence learning (GG), and a Bayesian LDA-based model using only observed context features to predict tag words (TJ). These sophisticated learning algorithms all outperform the traditional baseline of EM-HMM based methods,

\*This work is supported in part by the Lynn and William Frankel Center for Computer Science.

while relying on similar knowledge: the lexical context of the words to be tagged and their letter structure (e.g., presence of suffixes, capitalization and hyphenation).<sup>1</sup>

Our motivation for tackling unsupervised POS tagging is different: we are interested in developing a Hebrew POS tagger. We have access to a good Hebrew lexicon (and a morphological analyzer), and a fair amount of unlabeled training data, but hardly any annotated corpora. We actually report results on full morphological disambiguation for Hebrew, a task similar but more challenging than POS tagging: we deal with a tagset much larger than English (over 3,561 distinct tags) and an ambiguity level of about 2.7 per token as opposed to 1.4 for English. Instead of inventing a new learning framework, we go back to the traditional EM trained HMMs. We argue that the key challenge to learning an effective model is to define good enough initial conditions. Given sufficiently good initial conditions, EM trained models can yield highly competitive results. Such models have other benefits as well: they are simple, robust, and computationally more attractive.

In this paper, we concentrate on methods for deriving sufficiently good initial conditions for EM-HMM learning. Our method for learning initial conditions for the  $p(t|w)$  distributions relies on a mixture of language specific models: a paradigmatic model of similar words (where similar words are words with similar inflection patterns), simple syntagmatic constraints (e.g., the sequence V-V is extremely rare in English). These are complemented by a linear lexical context model. Such models are simple to build and test.

We present results for unsupervised PoS tagging of Hebrew text and for the common WSJ English test sets. We show that our method achieves state-of-the-art results for the English setting, even with a relatively small dictionary. Furthermore, while recent work report results on a reduced English tagset of 17 PoS tags, we also present results for the complete 45 tags tagset of the WSJ corpus. This considerably raises the bar of the EM-HMM baseline. We also report state-of-the-art results for Hebrew full mor-

phological disambiguation.

Our primary conclusion is that the problem of learning effective stochastic classifiers remains primarily a search task. Initial conditions play a dominant role in solving this task and can rely on linguistically motivated approximations. A robust learning method (EM-HMM) combined with good initial conditions based on a robust feature set can go a long way (as opposed to a more complex learning method). It seems that computing initial conditions is also the right place to capture complex linguistic intuition without fear that over-generalization could lead a learner to diverge.

## 2 Previous Work

The tagging accuracy of supervised stochastic taggers is around 96%–97% (Manning and Schutze, 1999). Merialdo (1994) reports an accuracy of 86.6% for an unsupervised token-based EM-estimated HMM, trained on a corpus of about 1M words, over a tagset of 159 tags. Elworthy (1994), in contrast, reports accuracy of 75.49%, 80.87%, and 79.12% for unsupervised word-based HMM trained on parts of the LOB corpora, with a tagset of 134 tags. With (artificially created) good initial conditions, such as a good approximation of the tag distribution for each word, Elworthy reports an improvement to 94.6%, 92.27%, and 94.51% on the same data sets. Merialdo, on the other hand, reports an improvement to 92.6% and 94.4% for the case where 100 and 2,000 sentences of the training corpus are manually tagged. Later, Banko and Moore (2004) observed that earlier unsupervised HMM-EM results were artificially high due to use of *Optimized Lexicons*, in which only frequent-enough analyses of each word were kept. Brill (1995b) proposed an unsupervised tagger based on transformation-based learning (Brill, 1995a), achieving accuracies of above 95%. This unsupervised tagger relied on an initial step in which the most probable tag for each word is chosen. Optimized lexicons and Brill’s most-probable-tag Oracle are not available in realistic unsupervised settings, yet, they show that *good initial conditions* greatly facilitate learning.

Recent work on unsupervised POS tagging for English has significantly improved the results on this task: GG, SE and most recently TJ report the best re-

<sup>1</sup>Another notable work, though within a slightly different framework, is the prototype-driven method proposed by (Haghighi and Klein, 2006), in which the dictionary is replaced with a very small seed of prototypical examples.

sults so far on the task of unsupervised POS tagging of the WSJ with diluted dictionaries. With dictionaries as small as 1249 lexical entries the LDA-based method with a strong ambiguity-class model reaches POS accuracy as high as 89.7% on a reduced tagset of 17 tags.

While these 3 methods rely on the same feature set (lexical context, spelling features) for the learning stage, the LDA approach bases its predictions entirely on observable features, and excludes the traditional hidden states sequence.

In Hebrew, Levinger *et al.* (1995) introduced the *similar-words algorithm* for estimating  $p(t|w)$  from unlabeled data, which we describe below. Our method uses this algorithm as a first step, and refines the approximation by introducing additional linguistic constraints and an iterative refinement step.

### 3 Initial Conditions For EM-HMM

The most common model for unsupervised learning of stochastic processes is Hidden Markov Models (HMM). For the case of tagging, the states correspond to the tags  $t_i$ , and words  $w_i$  are emitted each time a state is visited. The parameters of the model can be estimated by applying the Baum-Welch EM algorithm (Baum, 1972), on a large-scale corpus of unlabeled text. The estimated parameters are then used in conjunction with Viterbi search, to find the most probable sequence of tags for a given sentence. In this work, we follow Adler (2007) and use a variation of second-order HMM in which the probability of a tag is conditioned by the tag that precedes it and by the one that follows it, and the probability of an emitted word is conditioned by its tag and the tag that follows it<sup>2</sup>. In all experiments, we use the back-off smoothing method of (Thede and Harper, 1999), with additive smoothing (Chen, 1996) for the lexical probabilities.

We investigate methods to approximate the initial parameters of the  $p(t|w)$  distribution, from which we obtain  $p(w|t)$  by marginalization and Bayesian inversion. We also experiment with constraining the  $p(t|t_{-1}, t_{+1})$  distribution.

<sup>2</sup>Technically this is not Markov Model but a Dependency Net. However, bidirectional conditioning seem more suitable for language tasks, and in practice the learning and inference methods are mostly unaffected. See (Toutanova *et al.*, 2003).

**General syntagmatic constraints** We set linguistically motivated constraints on the  $p(t|t_{-1}, t_{+1})$  distribution. In our setting, these are used to force the probability of some events to 0 (e.g., “Hebrew verbs can not be followed by the *of* preposition”).

**Morphology-based  $p(t|w)$  approximation** Levinger *et al.* (1995) developed a context-free method for acquiring morpho-lexical probabilities ( $p(t|w)$ ) from an untagged corpus. The method is based on language-specific rules for constructing a *similar words* (SW) set for each analysis of a word. This set is composed of morphological variations of the word under the given analysis. For example, the Hebrew token ילד can be analyzed as either a noun (boy) or a verb (gave birth). The noun SW set for this token is composed of the definiteness and number inflections הילד, הילדים, הילדות (the boy, boys, the boys), while the verb SW set is composed of gender and tense inflections ילדה, ילדו (she/they gave birth). The approximated probability of each analysis is based on the corpus frequency of its SW set. For the complete details, refer to the original paper. Cucerzan and Yarowsky (2000) proposed a similar method for the unsupervised estimation of  $p(t|w)$  in English, relying on simple spelling features to characterize similar word classes.

**Linear-Context-based  $p(t|w)$  approximation** The method of Levinger *et al.* makes use of Hebrew inflection patterns in order to estimate context free approximation of  $p(t|w)$  by relating a word to its different inflections. However, the context in which a word occurs can also be very informative with respect to its POS-analysis (Schütze, 1995). We propose a novel algorithm for estimating  $p(t|w)$  based on the contexts in which a word occurs.<sup>3</sup>

The algorithm starts with an initial  $p(t|w)$  estimate, and iteratively re-estimates:

$$\hat{p}(t|c) = \frac{\sum_{w \in W} p(t|w)p(w|c)}{Z}$$

$$\hat{p}(t|w) = \frac{\sum_{c \in REL_C} p(t|c)p(c|w)allow(t, w)}{Z}$$

<sup>3</sup>While we rely on the same intuition, our use of context differs from earlier works on distributional POS-tagging like (Schütze, 1995), in which the purpose is to directly assign the possible POS for an unknown word. In contrast, our algorithm aims to improve the estimate for the whole distribution  $p(t|w)$ , to be further disambiguated by the EM-HMM learner.



where  $Z$  is a normalization factor,  $W$  is the set of all words in the corpus,  $C$  is the set of all contexts, and  $REL_C \subseteq C$  is a set of reliable contexts, defined below.  $allow(t, w)$  is a binary function indicating whether  $t$  is a valid tag for  $w$ .  $p(c|w)$  and  $p(w|c)$  are estimated via raw corpus counts.

Intuitively, we estimate the probability of a tag given a context as the average probability of a tag given any of the words appearing in that context, and similarly the probability of a tag given a word is the averaged probability of that tag in all the (reliable) contexts in which the word appears. At each round, we define  $REL_C$ , the set of reliable contexts, to be the set of all contexts in which  $p(t|c) > 0$  for at most  $X$  different  $ts$ .

The method is general, and can be applied to different languages. The parameters to specify for each language are: the initial estimation  $p(t|w)$ , the estimation of the  $allow$  relation for known and OOV words, and the types of contexts to consider.

## 4 Application to Hebrew

In Hebrew, several words combine into a single token in both agglutinative and fusional ways. This results in a potentially high number of tags for each token. On average, in our corpus, the number of possible analyses per known word reached 2.7, with the ambiguity level of the extended POS tagset in corpus for English (1.41) (Dermatas and Kokkinakis, 1995).

In this work, we use the morphological analyzer of MILA – Knowledge Center for Processing Hebrew (*KC analyzer*). In contrast to English tagsets, the number of tags for Hebrew, based on all combinations of the morphological attributes, can grow theoretically to about 300,000 tags. In practice, we found ‘only’ about 3,560 tags in a corpus of 40M tokens training corpus taken from Hebrew news material and Knesset transcripts. For testing, we manually tagged the text which is used in the Hebrew Treebank (Sima’an et al., 2001) (about 90K tokens), according to our tagging guidelines.

### 4.1 Initial Conditions

**General syntagmatic constraints** We define 4 syntagmatic constraints over  $p(t|t_{-1}, t_{+1})$ : (1) a construct state form cannot be followed by a verb,

preposition, punctuation, existential, modal, or copula; (2) a verb cannot be followed by the preposition  $\text{ל} \text{ } \text{של}$  (of), (3) copula and existential cannot be followed by a verb, and (4) a verb cannot be followed by another verb, unless one of them has a prefix, or the second verb is an infinitive, or the first verb is imperative and the second verb is in future tense.<sup>4</sup>

**Morphology-Based  $p(t|w)$  approximation** We extended the set of rules used in Levinger *et al.*, in order to support the wider tagset used by the KC analyzer: (1) The SW set for adjectives, copulas, existentials, personal pronouns, verbs and participles, is composed of all gender-number inflections; (2) The SW set for common nouns is composed of all number inflections, with definite article variation for absolute noun; (3) Prefix variations for proper nouns; (4) Gender variation for numerals; and (5) Gender-number variation for all suffixes (possessive, nominative and accusative).

### Linear-Context-based $p(t|w)$ approximation

For the initial  $p(t|w)$  we use either a uniform distribution based on the tags allowed in the dictionary, or the estimate obtained by using the modified Levinger *et al.* algorithm. We use contexts of the form  $L_R=w_{-1}, w_{+1}$  (the neighbouring words). We estimate  $p(w|c)$  and  $p(c|w)$  via relative frequency over all the events  $w1, w2, w3$  occurring at least 10 times in the corpus.  $allow(t, w)$  follows the dictionary. Because of the wide coverage of the Hebrew lexicon, we take  $REL_C$  to be  $C$  (all available contexts).

### 4.2 Evaluation

We run a series of experiments with 8 distinct initial conditions, as shown in Table 1: our baseline (*Uniform*) is the uniform distribution over all tags provided by the KC analyzer for each word. The *Syntagmatic* initial conditions add the  $p(t|t_{-1}, t_{+1})$  constraints described above to the uniform baseline. The *Morphology-Based* and *Linear-Context* initial conditions are computed as described above, while the *Morph+Linear* is the result of applying the linear-context algorithm over initial values computed by the Morphology-based method. We repeat

<sup>4</sup>This rule was taken from Shacham and Wintner(2007).

Initial Condition		Dist	Context-Free		EM-HMM	
			Full	Seg+Pos	Full	Seg+Pos
Uniform		60	63.8	71.9	85.5	89.8
Syntagmatic	Pair Constraints	60	/	/	85.8	89.8
	Init-Trans	60	/	/	87.9	91
Morpho-Lexical	Morph-Based	76.8	76.4	83.1	87.7	91.6
	Linear-Context	70.1	75.4	82.6	85.3	89.6
	Morph+Linear	<b>79.8</b>	<b>79.0</b>	<b>85.5</b>	88	92
PairConst+Morph	Morph-Based	/	/	/	87.6	91.4
	Linear-Context	/	/	/	84.5	89.0
	Morph+Linear	/	/	/	87.1	91.5
InitTrans+Morph	Morph-Based	/	/	/	89.2	92.3
	Linear-Context	/	/	/	87.7	90.9
	Morph+Linear	/	/	/	<b>89.4</b>	<b>92.4</b>

Table 1: Accuracy (%) of Hebrew Morphological Disambiguation and POS Tagging over various initial conditions

these last 3 models with the addition of the syntagmatic constraints (*Synt+Morph*).

For each of these, we first compare the computed  $p(t|w)$  against a gold standard distribution, taken from the test corpus (90K tokens), according to the measure used by (Levinger et al., 1995) (*Dist*). On this measure, we confirm that our improved morpho-lexical approximation improves the results reported by Levinger *et al.* from 74% to about 80% on a richer tagset, and on a much larger test set (90K vs. 3,400 tokens).

We then report on the effectiveness of  $p(t|w)$  as a context-free tagger that assigns to each word the most likely tag, both for full morphological analysis (3,561 tags) (*Full*) and for the simpler task of token segmentation and POS tag selection (36 tags) (*Seg+Pos*). The best results on this task are 80.8% and 87.5% resp. achieved on the *Morph+Linear* initial conditions.

Finally, we test effectiveness of the initial conditions with EM-HMM learning. We reach 88% accuracy on full morphological and 92% accuracy for POS tagging and word segmentation, for the *Morph+Linear* initial conditions.

As expected, EM-HMM improves results (from 80% to 88%). Strikingly, EM-HMM improves the uniform initial conditions from 64% to above 85%. However, better initial conditions bring us much over this particular local maximum – with an error reduction of 20%. In all cases, the main improvement over the uniform baseline is brought by the morphology-based initial conditions. When applied on its own, the linear context brings modest improvement. But the combination of the paradigmatic morphology-based method with the linear context

improves all measures.

A most interesting observation is the detrimental contribution of the syntagmatic constraints we introduced. We found that 113,453 sentences of the corpus (about 5%) contradict these basic and apparently simple constraints. As an alternative to these common-sense constraints, we tried to use a small seed of randomly selected sentences (10K annotated tokens) in order to skew the initial uniform distribution of the state transitions. We initialize the  $p(t|t_{-1}, t_{+1})$  distribution with smoothed ML estimates based on tag trigram and bigram counts (ignoring the tag-word annotations). This small seed initialization (*InitTrans*) has a great impact on accuracy. Overall, we reach 89.4% accuracy on full morphological and 92.4% accuracy for POS tagging and word segmentation, for the *Morph+Linear* conditions – an error reduction of more than 25% from the uniform distribution baseline.

## 5 Application to English

We now apply the same technique to English semi-supervised POS tagging. Recent investigations of this task use dictionaries derived from the Penn WSJ corpus, with a reduced tag set of 17 tags<sup>5</sup> instead of the original 45-tags tagset. They experiment with full dictionaries (containing complete POS information for all the words in the text) as well as “diluted” dictionaries, from which large portions of the vocabulary are missing. These settings are very different from those used for Hebrew: the tagset is much smaller (17 vs.  $\sim 3,560$ ) and the dictionaries are either complete or extremely crippled. However, for the sake of comparison, we have reproduced the same experimental settings.

We derive dictionaries from the complete WSJ corpus<sup>6</sup>, and the exact same diluted dictionaries used in SE, TJ and GG.

<sup>5</sup>ADJ ADV CONJ DET ENDPUNC INPUNC LPUNC RPUNC N POS PRT PREP PRT TO V VBG VBN WH

<sup>6</sup>The dictionary derived from the WSJ data is very noisy: many of the stop words get wrong analyses stemming from tagging mistakes (for instance, the word *the* has 6 possible analyses in the data-derived dictionary, which we checked manually and found all but DT erroneous). Such noise is not expected in a real world dictionary, and our algorithm is not designed to accommodate it. We corrected the entries for the 20 most frequent words in the corpus. This step could probably be done automatically, but we consider it to be a non-issue in any realistic setting.

**Syntagmatic Constraints** We indirectly incorporated syntagmatic constraints through a small change to the tagset. The 17-tags English tagset allows for V-V transitions. Such a construction is generally unlikely in English. By separating modals from the rest of the verbs, and creating an additional class for the 5 *be* verbs (am, is, are, was, were), we made such transition much less probable. The new 19-tags tagset reflects the “verb can not follow a verb” constraint.

**Morphology-Based  $p(t|w)$  approximation** English morphology is much simpler compared to that of Hebrew, making direct use of the Levinger context free approximation impossible. However, some morphological cues exist in English as well, in particular common suffixation patterns. We implemented our morphology-based context-free  $p(t|w)$  approximation for English as a special case of the linear context-based algorithm described in Sect.3. Instead of generating contexts based on neighboring words, we generate them using the following 5 morphological templates:

**suff=S** The word has suffix  $S$  (suff=ing).

**L+suff=W,S** The word appears just after word  $W$ , with suffix  $S$  (L+suff=have, ed).

**R+suff=S,W** The word appears just before word  $W$ , with suffix  $S$  (R+suff=ing, to)

**wsuf=S1,S2** The word suffix is  $S1$ , the same stem is seen with suffix  $S2$  (wsuf= $\epsilon$ , s).

**suffs=SG** The word stem appears with the  $SG$  group of suffixes (suffs=ed, ing, s).

We consider a word to have a suffix only if the word stem appears with a different suffix somewhere in the text. We implemented a primitive stemmer for extracting the suffixes while preserving a usable stem by taking care of few English orthography rules (handling, e.g., bigger  $\rightarrow$  big er, nicer  $\rightarrow$  nice er, happily  $\rightarrow$  happy ly, picnicking  $\rightarrow$  picnic ing). For the immediate context  $W$  in the templates  $L+suff, R+suff$ , we consider only the 20 most frequent tokens in the corpus.

**Linear-Context-based  $p(t|w)$  approximation** We expect the context based approximation to be particularly useful in English. We use the following 3 context templates:  $LL=w_{-2}, w_{-1}$ ,  $LR=w_{-1}, w_{+1}$  and  $RR=w_{+1}, w_{+2}$ . We estimate  $p(w|c)$  and  $p(c|w)$  by relative frequency over word triplets occurring at

least twice in the unannotated training corpus.

**Combined  $p(t|w)$  approximation** This approximation combines the morphological and linear context approximations by using all the above-mentioned context templates together in the iterative process.

For all three  $p(t|w)$  approximations, we take  $REL_C$  to be contexts containing at most 4 tags.  $allow(t, w)$  follows the dictionary for known words, and is the set of all open-class POS for unknown words. We take the initial  $p(t|w)$  for each  $w$  to be uniform over all the dictionary specified tags for  $w$ . Accordingly, the initial  $p(t|w) = 0$  for  $w$  not in the dictionary. We run the process for 8 iterations.<sup>7</sup>

### Diluted Dictionaries and Unknown Words

Some of the missing dictionary elements are assigned a set of possible POS-tags and corresponding probabilities in the  $p(t|w)$  estimation process. Other unknown tokens remain with no analysis at the end of the initial process computation. For these missing elements, we assign an ambiguity class by a simple ambiguity-class guesser, and set  $p(t|w)$  to be uniform over all the tags in the ambiguity class. Our ambiguity-class guesser assigns for each word the set of all open-class tags that appeared with the word suffix in the dictionary. The word suffix is the longest (up to 3 characters) suffix of the word that also appears in the top-100 suffixes in the dictionary.

**Taggers** We test the resulting  $p(t|w)$  approximation by training 2 taggers: **CF-Tag**, a context-free tagger assigning for each word its most probable POS according to  $p(t|w)$ , with a fallback to the most probable tag in case the word does not appear in the dictionary or if  $\forall t, p(t|w) = 0$ . **EM-HMM**, a second-order EM-HMM initialized with the estimated  $p(t|w)$ .

**Baselines** As baseline, we use two EM-trained HMM taggers, initialized with a uniform  $p(t|w)$  for every word, based on the allowed tags in the dictionary. For words not in the dictionary, we take the allowed tags to be either all the open-class POS

<sup>7</sup>This is the first value we tried, and it seems to work fine. We haven’t experimented with other values. The same applies for the choice of 4 as the  $REL_C$  threshold.

(**uniform(oc)**) or the allowed tags according to our simple ambiguity-class guesser (**uniform(suf)**).

All the  $p(t|w)$  estimates and HMM models are trained on the entire WSJ corpus. We use the same 24K word test-set as used in SE, TJ and GG, as well as the same diluted dictionaries. We report the results on the same reduced tagsets for comparison, but also include the results on the full 46 tags tagset.

## 5.1 Results

Table 2 summarizes the results of our experiments.

Uniform initialization based on the simple suffix-based ambiguity class guesser yields big improvements over the uniform all-open-class initialization. However, our refined initial conditions always improve the results (by as much as 40% error reduction). As expected, the linear context is much more effective than the morphological one, especially with richer dictionaries. This seem to indicate that in English the linear context is better at refining the estimations when the ambiguity classes are known, while the morphological context is in charge of adding possible tags when the ambiguity classes are not known. Furthermore, the benefit of the morphology-context is bigger for the complete tagset setting, indicating that, while the coarse-grained POS-tags are indicated by word distribution, the finer distinctions are indicated by inflections and orthography. The combination of linear and morphology contexts is always beneficial. Syntagmatic constraints (e.g., separating *be* verbs and modals from the rest of the verbs) constantly improve results by about 1%. Note that the context-free tagger based on our  $p(t|w)$  estimates is quite accurate. As with the EM trained models, combining linear and morphological contexts is always beneficial.

To put these numbers in context, Table 3 lists current state-of-the art results for the same task. **CE+spl** is the Contrastive-Estimation CRF method of SE. **BHMM** is the completely Bayesian-HMM of GG. **PLSA+AC**, **LDA**, **LDA+AC** are the models presented in TJ, LDA+AC is a Bayesian model with a strong ambiguity class (AC) component, and is the current state-of-the-art of this task. The other models are variations excluding the Bayesian components (PLSA+AC) or the ambiguity class.

While our models are trained on the unannotated text of the entire WSJ Treebank, CE and BHMM use

much less training data (only the 24k words of the test-set). However, as noted by TJ, there is no reason one should limit the amount of unlabeled data used, and in addition other results reported in GG,SE show that accuracy does not seem to improve as more unlabeled data are used with the models. We also report results for training our EM-HMM tagger on the smaller dataset (the  $p(t|w)$  estimation is still based on the entire unlabeled WSJ).

All the abovementioned models follow the assumption that all 17 tags are valid for the unknown words. In contrast, we restrict the set of allowed tags for an unknown word to open-class tags. Closed class words are expected to be included in a dictionary, even a small one. The practice of allowing only open-class tags for unknown words goes back a long way (Weischedel et al., 1993), and proved highly beneficial also in our case.

Notice that even our simplest models, in which the initial  $p(t|w)$  distribution for each  $w$  is uniform, already outperform most of the other models, and, in the case of the diluted dictionaries, by a wide margin. Similarly, given the  $p(t|w)$  estimate, EM-HMM training on the smaller dataset (24k) is still very competitive (yet results improve with more unlabeled data). When we use our refined  $p(t|w)$  distribution as the basis of EM-HMM training, we get the best results for the complete dictionary case. With the diluted dictionaries, we are outperformed only by LDA+AC. As we outperform this model in the complete dictionary case, it seems that the advantage of this model is due to its much stronger ambiguity class model, and not its Bayesian components. Also note that while we outperform this model when using the 19-tags tagset, it is slightly better in the original 17-tags setting. It could be that the reliance of the LDA models on observed surface features instead of hidden state features is beneficial avoiding the misleading V-V transitions.

We also list the performance of our best models with a slightly more realistic dictionary setting: we take our dictionary to include information for all words occurring in section 0-18 of the WSJ corpus (43208 words). We then train on the entire unannotated corpus, and test on sections 22-24 – the standard train/test split for supervised English POS tagging. We achieve accuracy of **92.85%** for the 19-tags set, and **91.3%** for the complete 46-tags tagset.

Initial Conditions	Full dict (49206 words)		$\geq 2$ dict (2141 words)		$\geq 3$ dict (1249 words)		
	CF-Tag	EM-HMM	CF-Tag	EM-HMM	CF-Tag	EM-HMM	
17tags	Uniform(oc)	81.7	88.7	68.4	81.9	62.5	79.6
	Uniform(suf)	NA	NA	76.8	83.4	76.9	81.6
	Morph-Cont	82.2	88.6	73.3	83.9	69.1	81.7
	Linear-Cont	90.1	92.9	81.1	87.8	78.3	85.8
	Combined-Cont	89.9	93.3	83.1	88.5	81.1	86.4
19tags	Uniform(oc)	79.9	91.0	66.6	83.4	60.7	84.7
	Uniform(suf)	NA	NA	75.1	86.5	73.1	86.7
	Morph-Cont	80.5	89.2	71.5	86.5	67.5	87.1
	Linear-Cont	88.4	93.7	78.9	89.0	76.3	86.9
	Combined-Cont	88.0	93.8	81.1	89.4	79.2	87.4
46tags	Uniform(oc)	76.7	88.3	61.2	*	55.7	*
	Uniform(suf)	NA	NA	64.2	81.9	60.3	79.8
	Morph-Cont	74.8	88.8	65.6	83.0	61.9	80.3
	Linear-Cont	85.5	91.2	74.5	84.0	70.1	82.2
	Combined-Cont	85.9	91.4	75.4	85.5	72.4	83.3

Table 2: Accuracy (%) of English POS Tagging over various initial conditions

Dict	InitEM-HMM (24k)	LDA	LDA+AC	PLSA+AC	CE+spl	BHMM
Full	<b>93.8</b> (91.1)	93.4	93.4	89.7	88.7	87.3
$\geq 2$	89.4 (87.9)	87.4	<b>91.2</b>	87.8	79.5	79.6
$\geq 3$	87.4 (85.9)	85	<b>89.7</b>	85.9	78.4	71

Table 3: Comparison of English Unsupervised POS Tagging Methods

## 6 Conclusion

We have demonstrated that unsupervised POS tagging can reach good results using the robust EM-HMM learner when provided with good initial conditions, even with incomplete dictionaries. We presented a general family of algorithms to compute effective initial conditions: estimation of  $p(t|w)$  relying on an iterative process shifting probabilities between words and their contexts. The parameters of this process (definition of the contexts and initial estimations of  $p(t|w)$ ) can safely encapsulate rich linguistic intuitions.

While recent work, such as GG, aim to use the Bayesian framework and incorporate “linguistically motivated priors”, in practice such priors currently only account for the fact that language related distributions are sparse - a very general kind of knowledge. In contrast, our method allow the incorporation of much more fine-grained intuitions.

We tested the method on the challenging task of full morphological disambiguation in Hebrew (which was our original motivation) and on the standard WSJ unsupervised POS tagging task.

In Hebrew, our model includes an improved version of the *similar words* algorithm of (Levinger et al., 1995), a model of lexical context, and a small

set of tag ngrams. The combination of these knowledge sources in the initial conditions brings an error reduction of more than 25% over a strong uniform distribution baseline. In English, our model is competitive with recent state-of-the-art results, while using simple and efficient learning methods.

The comparison with other algorithms indicates directions of potential improvement: (1) our initial-conditions method might benefit the other, more sophisticated learning algorithms as well. (2) Our models were designed under the assumption of a relatively complete dictionary. As such, they are not very good at assigning ambiguity-classes to OOV tokens when starting with a very small dictionary. While we demonstrate competitive results using a simple suffix-based ambiguity-class guesser which ignores capitalization and hyphenation information, we believe there is much room for improvement in this respect. In particular, (Haghighi and Klein, 2006) presents very strong results using a distributional-similarity module and achieve impressive tagging accuracy while starting with a mere 116 prototypical words. Experimenting with combining similar models (as well as TJ’s ambiguity class model) with our  $p(t|w)$  distribution estimation method is an interesting research direction.

## References

- Meni Adler. 2007. *Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach*. Ph.D. thesis, Ben-Gurion University of the Negev, Beer-Sheva, Israel.
- Michele Banko and Robert C. Moore. 2004. Part-of-speech tagging in context. In *Proceedings of Coling 2004*, pages 556–561, Geneva, Switzerland, Aug 23–Aug 27. COLING.
- Leonard E. Baum. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process. *Inequalities*, 3:1–8.
- Eric Brill. 1995a. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21:543–565.
- Eric Brill. 1995b. Unsupervised learning of disambiguation rules for part of speech tagging. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 1–13, Somerset, New Jersey. Association for Computational Linguistics.
- Stanley F. Chen. 1996. *Building Probabilistic Models for Natural Language*. Ph.D. thesis, Harvard University, Cambridge, MA.
- Silviu Cucerzan and David Yarowsky. 2000. Language independent, minimally supervised induction of lexical probabilities. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 270–277, Morristown, NJ, USA. Association for Computational Linguistics.
- Evangelos Dermatas and George Kokkinakis. 1995. Automatic stochastic tagging of natural language texts. *Computational Linguistics*, 21(2):137–163.
- David Elworthy. 1994. Does Baum-Welch re-estimation help taggers? In *Proceeding of ANLP-94*.
- Sharon Goldwater and Thomas L. Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceeding of ACL 2007*, Prague, Czech Republic.
- Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 320–327, Morristown, NJ, USA. Association for Computational Linguistics.
- J. Kupiec. 1992. Robust part-of-speech tagging using hidden Markov model. *Computer Speech and Language*, 6:225–242.
- Moshe Lévinger, Uzi Ornan, and Alon Itai. 1995. Learning morpholexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics*, 21:383–404.
- Christopher D. Manning and Hinrich Schütze. 1999. *Foundation of Statistical Language Processing*. MIT Press.
- Bernard Merialdo. 1994. Tagging English text with probabilistic model. *Computational Linguistics*, 20:155–171.
- Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 141–148, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Danny Shacham and Shuly Wintner. 2007. Morphological disambiguation of hebrew: A case study in classifier combination. In *Proceeding of EMNLP-07*, Prague, Czech.
- Khalil Sima'an, Alon Itai, Alon Altman Yoad Winter, and Noa Nativ. 2001. Building a tree-bank of modern Hebrew text. *Journal Traitement Automatique des Langues (t.a.l.)*. Special Issue on NLP and Corpus Linguistics.
- Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 354–362, Ann Arbor, Michigan, June.
- Scott M. Theede and Mary P. Harper. 1999. A second-order hidden Markov model for part-of-speech tagging. In *Proceeding of ACL-99*.
- Kristina Toutanova and Mark Johnson. 2008. A bayesian lda-based model for semi-supervised part-of-speech tagging. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*.
- R. Weischedel, R. Schwartz, J. Palmucci, M. Meteor, and L. Ramshaw. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19:359–382.

# Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation

Jakob Uszkoreit\* Thorsten Brants

Google, Inc.  
1600 Amphitheatre Parkway  
Mountain View, CA 94303, USA  
{uszkoreit,brants}@google.com

## Abstract

In statistical language modeling, one technique to reduce the problematic effects of data sparsity is to partition the vocabulary into equivalence classes. In this paper we investigate the effects of applying such a technique to higher-order  $n$ -gram models trained on large corpora. We introduce a modification of the exchange clustering algorithm with improved efficiency for certain partially class-based models and a distributed version of this algorithm to efficiently obtain automatic word classifications for large vocabularies (>1 million words) using such large training corpora (>30 billion tokens). The resulting clusterings are then used in training partially class-based language models. We show that combining them with word-based  $n$ -gram models in the log-linear model of a state-of-the-art statistical machine translation system leads to improvements in translation quality as indicated by the BLEU score.

## 1 Introduction

A statistical language model assigns a probability  $P(w)$  to any given string of words  $w_1^m = w_1, \dots, w_m$ . In the case of  $n$ -gram language models this is done by factoring the probability:

$$P(w_1^m) = \prod_{i=1}^m P(w_i | w_1^{i-1})$$

and making a Markov assumption by approximating this by:

$$\prod_{i=1}^m P(w_i | w_1^{i-1}) \approx \prod_{i=1}^m p(w_i | w_{i-n+1}^{i-1})$$

Even after making the Markov assumption and thus treating all strings of preceding words as equal which

\* Parts of this research were conducted while the author studied at the Berlin Institute of Technology

do not differ in the last  $n - 1$  words, one problem  $n$ -gram language models suffer from is that the training data is too sparse to reliably estimate all conditional probabilities  $P(w_i | w_1^{i-1})$ .

Class-based  $n$ -gram models are intended to help overcome this data sparsity problem by grouping words into equivalence classes rather than treating them as distinct words and thus reducing the number of parameters of the model (Brown et al., 1990). They have often been shown to improve the performance of speech recognition systems when combined with word-based language models (Martin et al., 1998; Whittaker and Woodland, 2001). However, in the area of statistical machine translation, especially in the context of large training corpora, fewer experiments with class-based  $n$ -gram models have been performed with mixed success (Raab, 2006).

Class-based  $n$ -gram models have also been shown to benefit from their reduced number of parameters when scaling to higher-order  $n$ -grams (Goodman and Gao, 2000), and even despite the increasing size and decreasing sparsity of language model training corpora (Brants et al., 2007), class-based  $n$ -gram models might lead to improvements when increasing the  $n$ -gram order.

When training class-based  $n$ -gram models on large corpora and large vocabularies, one of the problems arising is the scalability of the typical clustering algorithms used for obtaining the word classification. Most often, variants of the *exchange algorithm* (Kneser and Ney, 1993; Martin et al., 1998) or the agglomerative clustering algorithm presented in (Brown et al., 1990) are used, both of which have prohibitive runtimes when clustering large vocabularies on the basis of large training corpora with a sufficiently high number of classes.

In this paper we introduce a modification of the exchange algorithm with improved efficiency and then present a distributed version of the modified algorithm, which makes it feasible to obtain word clas-

sifications using billions of tokens of training data. We then show that using partially class-based language models trained using the resulting classifications together with word-based language models in a state-of-the-art statistical machine translation system yields improvements despite the very large size of the word-based models used.

## 2 Class-Based Language Modeling

By partitioning all  $N_v$  words of the vocabulary into  $N_c$  sets, with  $c(w)$  mapping a word onto its equivalence class and  $c(w_i^j)$  mapping a sequence of words onto the sequence of their respective equivalence classes, a typical class-based  $n$ -gram model approximates  $P(w_i|w_1^{i-1})$  with the two following component probabilities:

$$P(w_i|w_1^{i-1}) \approx p_0(w_i|c(w_i)) \cdot p_1(c(w_i)|c(w_{i-n+1}^{i-1})) \quad (1)$$

thus greatly reducing the number of parameters in the model, since usually  $N_c$  is much smaller than  $N_v$ .

In the following, we will call this type of model a *two-sided class-based model*, as both the history of each  $n$ -gram, the sequence of words conditioned on, as well as the predicted word are replaced by their class.

Once a partition of the words in the vocabulary is obtained, two-sided class-based models can be built just like word-based  $n$ -gram models using existing infrastructure. In addition, the size of the model is usually greatly reduced.

### 2.1 One-Sided Class-Based Models

Two-sided class-based models received most attention in the literature. However, several different types of mixed word and class models have been proposed for the purpose of improving the performance of the model (Goodman, 2000), reducing its size (Goodman and Gao, 2000) as well as lowering the complexity of related clustering algorithms (Whittaker and Woodland, 2001).

In (Emami and Jelinek, 2005) a clustering algorithm is introduced which outputs a separate clustering for each word position in a trigram model. In the experimental evaluation, the authors observe the largest improvements using a specific clustering for the last word of each trigram but no clustering at all for the first two word positions. Generalizing this leads to arbitrary order class-based  $n$ -gram models of the form:

$$P(w_i|w_1^{i-1}) \approx p_0(w_i|c(w_i)) \cdot p_1(c(w_i)|w_{i-n+1}^{i-1}) \quad (2)$$

which we will call *predictive class-based models* in the following sections.

## 3 Exchange Clustering

One of the frequently used algorithms for automatically obtaining partitions of the vocabulary is the exchange algorithm (Kneser and Ney, 1993; Martin et al., 1998). Beginning with an initial clustering, the algorithm greedily maximizes the log likelihood of a two-sided class bigram or trigram model as described in Eq. (1) on the training data. Let  $V$  be the set of words in the vocabulary and  $C$  the set of classes. This then leads to the following optimization criterion, where  $N(w)$  and  $N(c)$  denote the number of occurrences of a word  $w$  or a class  $c$  in the training data and  $N(c, d)$  denotes the number of occurrences of some word in class  $c$  followed by a word in class  $d$  in the training data:

$$\begin{aligned} \hat{C} = \operatorname{argmax}_C & \sum_{w \in V} N(w) \cdot \log N(w) + \\ & + \sum_{c \in C, d \in C} N(c, d) \cdot \log N(c, d) - \\ & - 2 \cdot \sum_{c \in C} N(c) \cdot \log N(c) \end{aligned} \quad (3)$$

The algorithm iterates over all words in the vocabulary and tentatively moves each word to each cluster. The change in the optimization criterion is computed for each of these tentative moves and the exchange leading to the highest increase in the optimization criterion (3) is performed. This procedure is then repeated until the algorithm reaches a local optimum.

To be able to efficiently calculate the changes in the optimization criterion when exchanging a word, the counts in Eq. (3) are computed once for the initial clustering, stored, and afterwards updated when a word is exchanged.

Often only a limited number of iterations are performed, as letting the algorithm terminate in a local optimum can be computationally impractical.

### 3.1 Complexity

The implementation described in (Martin et al., 1998) uses a memory saving technique introducing a binary search into the complexity estimation. For the sake of simplicity, we omit this detail in the following complexity analysis. We also do not employ this optimization in our implementation.

The worst case complexity of the exchange algorithm is quadratic in the number of classes. However,



**Input:** The fixed number of clusters  $N_c$   
 Compute initial clustering  
**while** *clustering changed in last iteration* **do**  
   **forall**  $w \in V$  **do**  
     **forall**  $c \in C$  **do**  
       move word  $w$  tentatively to cluster  
        $c$   
       compute updated optimization  
       criterion  
     move word  $w$  to cluster maximizing  
     optimization criterion

**Algorithm 1:** Exchange Algorithm Outline

the average case complexity can be reduced by updating only the counts which are actually affected by moving a word from one cluster to another. This can be done by considering only those sets of clusters for which  $N(w, c) > 0$  or  $N(c, w) > 0$  for a word  $w$  about to be exchanged, both of which can be calculated efficiently when exchanging a word. The algorithm scales linearly in the size of the vocabulary.

With  $N_c^{pre}$  and  $N_c^{suc}$  denoting the average number of clusters preceding and succeeding another cluster,  $B$  denoting the number of distinct bigrams in the training corpus, and  $I$  denoting the number of iterations, the worst case complexity of the algorithm is in:

$$\mathcal{O}(I \cdot (2 \cdot B + N_v \cdot N_c \cdot (N_c^{pre} + N_c^{suc})))$$

When using large corpora with large numbers of bigrams the number of required updates can increase towards the quadratic upper bound as  $N_c^{pre}$  and  $N_c^{suc}$  approach  $N_c$ . For a more detailed description and further analysis of the complexity, the reader is referred to (Martin et al., 1998).

## 4 Predictive Exchange Clustering

Modifying the exchange algorithm in order to optimize the log likelihood of a predictive class bigram model, leads to substantial performance improvements, similar to those previously reported for another type of one-sided class model in (Whittaker and Woodland, 2001).

We use a predictive class bigram model as given in Eq. (2), for which the maximum-likelihood probability estimates for the  $n$ -grams are given by their relative frequencies:

$$\begin{aligned} P(w_i | w_1^{i-1}) &\approx p_0(w_i | c(w_i)) \cdot p_1(c(w_i) | w_{i-1}) \quad (4) \\ &= \frac{N(w_i)}{N(c(w_i))} \cdot \frac{N(w_{i-1}, c(w_i))}{N(w_{i-1})} \quad (5) \end{aligned}$$

where  $N(w)$  again denotes the number of occurrences of the word  $w$  in the training corpus and  $N(v, c)$

the number of occurrences of the word  $v$  followed by some word in class  $c$ . Then the following optimization criterion can be derived, with  $F(C)$  being the log likelihood function of the predictive class bigram model given a clustering  $C$ :

$$\begin{aligned} F(C) &= \sum_{w \in V} N(w) \cdot \log p(w | c(w)) \\ &\quad + \sum_{v \in V, c \in C} N(v, c) \cdot \log p(c | v) \quad (6) \end{aligned}$$

$$\begin{aligned} &= \sum_{w \in V} N(w) \cdot \log \frac{N(w)}{N(c(w))} \\ &\quad + \sum_{v \in V, c \in C} N(v, c) \cdot \log \frac{N(v, c)}{N(v)} \quad (7) \end{aligned}$$

$$\begin{aligned} &= \sum_{w \in V} N(w) \cdot \log N(w) \\ &\quad - \sum_{w \in V} N(w) \cdot \log N(c(w)) \\ &\quad + \sum_{v \in V, c \in C} N(v, c) \cdot \log N(v, c) \\ &\quad - \sum_{v \in V, c \in C} N(v, c) \cdot \log N(v) \quad (8) \end{aligned}$$

The very last summation of Eq. (8) now effectively sums over all occurrences of all words and thus cancels out with the first summation of (8) which leads to:

$$\begin{aligned} F(C) &= \sum_{v \in V, c \in C} N(v, c) \cdot \log N(v, c) \\ &\quad - \sum_{w \in V} N(w) \cdot \log N(c(w)) \quad (9) \end{aligned}$$

In the first summation of Eq. (9), for a given word  $v$  only the set of classes which contain at least one word  $w$  for which  $N(v, w) > 0$  must be considered, denoted by  $suc(v)$ . The second summation is equivalent to  $\sum_{c \in C} N(c) \cdot \log N(c)$ . Thus the further simplified criterion is:

$$\begin{aligned} F(C) &= \sum_{v \in V, c \in suc(v)} N(v, c) \cdot \log N(v, c) \\ &\quad - \sum_{c \in C} N(c) \cdot \log N(c) \quad (10) \end{aligned}$$

When exchanging a word  $w$  between two classes  $c$  and  $c'$ , only two summands of the second summation of Eq. (10) are affected. The first summation can be updated by iterating over all bigrams ending in the exchanged word. Throughout one iteration of the algorithm, in which for each word in the vocabulary each possible move to another class is evaluated, this

amounts to the number of distinct bigrams in the training corpus  $B$ , times the number of clusters  $N_c$ . Thus the worst case complexity using the modified optimization criterion is in:

$$\mathcal{O}(I \cdot N_c \cdot (B + N_v))$$

Using this optimization criterion has two effects on the complexity of the algorithm. The first difference is that in contrast to the exchange algorithm using a two sided class-based bigram model in its optimization criterion, only two clusters are affected by moving a word. Thus the algorithm scales linearly in the number of classes. The second difference is that  $B$  dominates the term  $B + N_v$  for most corpora and scales far less than linearly with the vocabulary size, providing a significant performance advantage over the other optimization criterion, especially when large vocabularies are used (Whittaker and Woodland, 2001).

For efficiency reasons, an exchange of a word between two clusters is separated into a *remove* and a *move* procedure. In each iteration the *remove* procedure only has to be called once for each word, while for a given word *move* is called once for every cluster to compute the consequences of the tentative exchanges. An outline of the *move* procedure is given below. The *remove* procedure is similar.

**Input:** A word  $w$ , and a destination cluster  $c$

**Result:** The change in the optimization criterion when moving  $w$  to cluster  $c$

$\delta \leftarrow N(c) \cdot \log N(c)$

$N'(c) \leftarrow N(c) - N(w)$

$\delta \leftarrow \delta - N'(c) \cdot \log N'(c)$

**if not a tentative move then**

$N(c) \leftarrow N'(c)$

**forall**  $v \in \text{suc}(w)$  **do**

$\delta \leftarrow \delta - N(v, c) \cdot \log N(v, c)$

$N'(v, c) \leftarrow N(v, c) - N(v, w)$

$\delta \leftarrow \delta + N'(v, c) \cdot \log N'(v, c)$

**if not a tentative move then**

$N(v, c) \leftarrow N'(v, c)$

**return**  $\delta$

**Procedure MoveWord**

## 5 Distributed Clustering

When training on large corpora, even the modified exchange algorithm would still require several days if not weeks of CPU time for a sufficient number of iterations.

To overcome this we introduce a novel *distributed exchange algorithm*, based on the modified exchange

algorithm described in the previous section. The vocabulary is randomly partitioned into sets of roughly equal size. With each word  $w$  in one of these sets, all words  $v$  preceding  $w$  in the corpus are stored with the respective bigram count  $N(v, w)$ .

The clusterings generated in each iteration as well as the initial clustering are stored as the set of words in each cluster, the total number of occurrences of each cluster in the training corpus, and the list of words preceding each cluster. For each word  $w$  in the predecessor list of a given cluster  $c$ , the number of times  $w$  occurs in the training corpus before any word in  $c$ ,  $N(w, c)$ , is also stored.

Together with the counts stored with the vocabulary partitions, this allows for efficient updating of the terms in Eq. (10).

The initial clustering together with all the required counts is created in an initial iteration by assigning the  $n$ -th most frequent word to cluster  $n \bmod N_c$ . While (Martin et al., 1998) and (Emami and Jelinek, 2005) observe that the initial clustering does not seem to have a noticeable effect on the quality of the resulting clustering or the convergence rate, the intuition behind this method of initialization is that it is unlikely for the most frequent words to be clustered together due to their high numbers of occurrences.

In each subsequent iteration each one of a number of workers is assigned one of the partitions of the words in the vocabulary. After loading the current clustering, it then randomly chooses a subset of these words of a fixed size. For each of the selected words the worker then determines to which cluster the word is to be moved in order to maximize the increase in log likelihood, using the count updating procedures described in the previous section. All changes a worker makes to the clustering are accumulated locally in delta data structures. At the end of the iteration all deltas are merged and applied to the previous clustering, resulting in the complete clustering loaded in the next iteration.

This algorithm fits well into the *MapReduce* programming model (Dean and Ghemawat, 2004) that we used for our implementation.

### 5.1 Convergence

While the greedy non-distributed exchange algorithm is guaranteed to converge as each exchange increases the log likelihood of the assumed bigram model, this is not necessarily true for the distributed exchange algorithm. This stems from the fact that the change in log likelihood is calculated by each worker under the assumption that no other changes to the clustering are performed by other workers in

this iteration. However, if in each iteration only a rather small and randomly chosen subset of all words are considered for exchange, the intuition is that the remaining words still define the parameters of each cluster well enough for the algorithm to converge.

In (Emami and Jelinek, 2005) the authors observe that only considering a subset of the vocabulary of half the size of the complete vocabulary in each iteration does not affect the time required by the exchange algorithm to converge. Yet each iteration is sped up by approximately a factor of two. The quality of class-based models trained using the resulting clusterings did not differ noticeably from those trained using clusterings for which the full vocabulary was considered in each iteration. Our experiments showed that this also seems to be the case for the distributed exchange algorithm. While considering very large subsets of the vocabulary in each iteration can cause the algorithm to not converge at all, considering only a very small fraction of the words for exchange will increase the number of iterations required to converge. In experiments we empirically determined that choosing a subset of roughly a third of the size of the full vocabulary is a good balance in this trade-off. We did not observe the algorithm to not converge unless we used fractions above half of the vocabulary size.

We typically ran the clustering for 20 to 30 iterations after which the number of words exchanged in each iteration starts to stabilize at less than 5 percent of the vocabulary size. Figure 1 shows the number of words exchanged in each of 34 iterations when clustering the approximately 300,000 word vocabulary of the Arabic side of the English-Arabic parallel training data into 512 and 2,048 clusters.

Despite a steady reduction in the number of words exchanged per iteration, we observed the convergence in regards to log-likelihood to be far from monotone. In our experiments we were able to achieve significantly more monotone and faster convergence by employing the following heuristic. As described in Section 5, we start out the first iteration with a random partition of the vocabulary into subsets each assigned to a specific worker. However, instead of keeping this assignment constant throughout all iterations, after each iteration the vocabulary is partitioned anew so that all words from any given cluster are considered by the same worker in the next iteration. The intuition behind this heuristic is that as the clustering becomes more coherent, the information each worker has about groups of similar words is becoming increasingly accurate. In our experiments this heuristic lead to almost monotone convergence in log-likelihood. It also reduced the

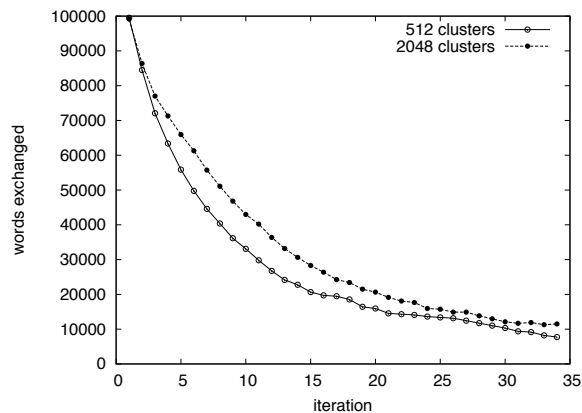


Figure 1: Number of words exchanged per iteration when clustering the vocabulary of the Arabic side of the English-Arabic parallel training data (347 million tokens).

number of iterations required to converge by up to a factor of three.

## 5.2 Resource Requirements

The runtime of the distributed exchange algorithm depends highly on the number of distinct bigrams in the training corpus. When clustering the approximately 1.5 million word vocabulary of a 405 million token English corpus into 1,000 clusters, one iteration takes approximately 5 minutes using 50 workers based on standard hardware running the Linux operating system. When clustering the 0.5 million most frequent words in the vocabulary of an English corpus with 31 billion tokens into 1,000 clusters, one iteration takes approximately 30 minutes on 200 workers.

When scaling up the vocabulary and corpus sizes, the current bottleneck of our implementation is loading the current clustering into memory. While the memory requirements decrease with each iteration, during the first few iterations a worker typically still needs approximately 2 GB of memory to load the clustering generated in the previous iteration when training 1,000 clusters on the 31 billion token corpus.

## 6 Experiments

We trained a number of predictive class-based language models on different Arabic and English corpora using clusterings trained on the complete data of the same corpus. We use the distributed training and application infrastructure described in (Brants et al., 2007) with modifications to allow the training of predictive class-based models and their application in the decoder of the machine translation system.

For all models used in our experiments, both word- and class-based, the smoothing method used was *Stupid Backoff* (Brants et al., 2007). Models with Stupid Backoff return scores rather than normalized probabilities, thus perplexities cannot be calculated for these models. Instead we report BLEU scores (Papineni et al., 2002) of the machine translation system using different combinations of word- and class-based models for translation tasks from English to Arabic and Arabic to English.

## 6.1 Training Data

For English we used three different training data sets: ***en\_target***: The English side of Arabic-English and Chinese-English parallel data provided by LDC (405 million tokens).

***en\_ldcnews***: Consists of several English news data sets provided by LDC (5 billion tokens).

***en\_webnews***: Consists of data collected up to December 2005 from web pages containing primarily English news articles (31 billion tokens).

A fourth data set, *en\_web*, was used together with the other three data sets to train the large word-based model used in the second machine translation experiment. This set consists of general web data collected in January 2006 (2 trillion tokens).

For Arabic we used the following two different training data sets:

***ar\_gigaword***: Consists of several Arabic news data sets provided by LDC (629 million tokens).

***ar\_webnews***: Consists of data collected up to December 2005 from web pages containing primarily Arabic news articles (approximately 600 million tokens).

## 6.2 Machine Translation Results

Given a sentence  $\mathbf{f}$  in the source language, the *machine translation* problem is to automatically produce a translation  $\hat{\mathbf{e}}$  in the target language. In the subsequent experiments, we use a phrase-based statistical machine translation system based on the log-linear formulation of the problem described in (Och and Ney, 2002):

$$\begin{aligned} \hat{\mathbf{e}} &= \operatorname{argmax}_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \\ &= \operatorname{argmax}_{\mathbf{e}} \sum_{m=1}^M \lambda_m h_m(\mathbf{e}, \mathbf{f}) \end{aligned} \quad (11)$$

where  $\{h_m(\mathbf{e}, \mathbf{f})\}$  is a set of  $M$  *feature functions* and  $\{\lambda_m\}$  a set of *weights*. We use each predictive class-based language model as well as a word-based model as separate feature functions in the log-linear combination in Eq. (11). The weights are trained using

minimum error rate training (Och, 2003) with BLEU score as the objective function.

The *dev* and *test* data sets contain parts of the 2003, 2004 and 2005 Arabic NIST MT evaluation sets among other parallel data. The blind test data used is the “NIST” part of the 2006 Arabic-English NIST MT evaluation set, and is not included in the training data.

For the first experiment we trained predictive class-based 5-gram models using clusterings with 64, 128, 256 and 512 clusters<sup>1</sup> on the *en\_target* data. We then added these models as additional features to the log linear model of the Arabic-English machine translation system. The word-based language model used by the system in these experiments is a 5-gram model also trained on the *en\_target* data set.

Table 1 shows the BLEU scores reached by the translation system when combining the different class-based models with the word-based model in comparison to the BLEU scores by a system using only the word-based model on the Arabic-English translation task.

	<b>dev</b>	<b>test</b>	<b>nist06</b>
<b>word-based only</b>	0.4085	0.3498	0.5088
<b>64 clusters</b>	0.4122	0.3514	0.5114
<b>128 clusters</b>	0.4142	0.3530	0.5109
<b>256 clusters</b>	0.4141	0.3536	0.5076
<b>512 clusters</b>	0.4120	0.3504	0.5140

Table 1: BLEU scores of the Arabic English system using models trained on the English *en\_target* data set

Adding the class-based models leads to small improvements in BLEU score, with the highest improvements for both *dev* and *nist06* being statistically significant<sup>2</sup>.

In the next experiment we used two predictive class-based models, a 5-gram model with 512 clusters trained on the *en\_target* data set and a 6-gram model also using 512 clusters trained on the *en\_ldcnews* data set. We used these models in addition to a word-based 6-gram model created by combining models trained on all four English data sets.

Table 2 shows the BLEU scores of the machine translation system using only this word-based model, the scores after adding the class-based model trained on the *en\_target* data set and when using all three models.

<sup>1</sup>The *beginning of sentence*, *end of sentence* and *unknown word* tokens were each treated as separate clusters

<sup>2</sup>Differences of more than 0.0051 are statistically significant at the 0.05 level using bootstrap resampling (Noreen, 1989; Koehn, 2004)

	dev	test	nist06
<b>word-based only</b>	0.4677	0.4007	0.5672
<b>with <i>en_target</i></b>	0.4682	0.4022	0.5707
<b>all three models</b>	0.4690	0.4059	0.5748

Table 2: BLEU scores of the Arabic English system using models trained on various data sets

For our experiment with the English Arabic translation task we trained two 5-gram predictive class-based models with 512 clusters on the Arabic *ar\_gigaword* and *ar\_webnews* data sets. The word-based Arabic 5-gram model we used was created by combining models trained on the Arabic side of the parallel training data (347 million tokens), the *ar\_gigaword* and *ar\_webnews* data sets, and additional Arabic web data.

	dev	test	nist06
<b>word-based only</b>	0.2207	0.2174	0.3033
<b>with <i>ar_webnews</i></b>	0.2237	0.2136	0.3045
<b>all three models</b>	0.2257	0.2260	0.3318

Table 3: BLEU scores of the English Arabic system using models trained on various data sets

As shown in Table 3, adding the predictive class-based model trained on the *ar\_webnews* data set leads to small improvements in *dev* and *nist06* scores but causes the *test* score to decrease. However, adding the class-based model trained on the *ar\_gigaword* data set to the other class-based and the word-based model results in further improvement of the *dev* score, but also in large improvements of the *test* and *nist06* scores.

We performed experiments to eliminate the possibility of data overlap between the training data and the machine translation test data as cause for the large improvements. In addition, our experiments showed that when there is overlap between the training and test data, the class-based models lead to lower scores as long as they are trained only on data also used for training the word-based model. One explanation could be that the domain of the *ar\_gigaword* corpus is much closer to the domain of the test data than that of other training data sets used. However, further investigation is required to explain the improvements.

### 6.3 Clusters

The clusters produced by the distributed algorithm vary in their size and number of occurrences. In a clustering of the *en\_target* data set with 1,024 clusters, the cluster sizes follow a typical long-tailed distribution with the smallest cluster contain-

Bai Bi Bu Cai Cao Chang Chen Cheng Chou Chuang Cui Dai Deng Ding Du Duan Fan Fu Gao Ge Geng Gong Gu Guan Han Hou Hsiao Hsieh Hsu Hu Huang Huo Jiang Jiao Juan Kang Kuang Kuo Li Liang Liao Lin Liu Lu Luo Mao Meets Meng Mi Miao Mu Niu Pang Pi Pu Qian Qiao Qiu Qu Ren Run Shan Shang Shen Si Song Su Sui Sun Tan Tang Tian Tu Wang Wu Xie Xiong Xu Yang Yao Ye Yin Zeng Zhang Zhao Zheng Zhou Zhu Zhuang Zou
% PERCENT cents percent
approvals bonus cash concessions cooperatives credit disbursements dividends donations earnings emoluments entitlements expenditure expenditures fund funding funds grants income incomes inflation lending liquidity loan loans mortgage mortgages overhead payroll pension pensions portfolio profits protectionism quotas receipts receivables remittances remuneration rent rents returns revenue revenues salaries salary savings spending subscription subsidies subsidy surplus surpluses tax taxation taxes tonnage tuition turnover wage wages
Abby Abigail Agnes Alexandra Alice Amanda Amy Andrea Angela Ann Anna Anne Annette Becky Beth Betsy Bonnie Brenda Carla Carol Carole Caroline Carolyn Carrie Catherine Cathy Cheryl Christina Christine Cindy Claire Clare Claudia Colleen Cristina Cynthia Danielle Daphne Dawn Debbie Deborah Denise Diane Dina Dolores Donna Doris Edna Eileen Elaine Eleanor Elena Elisabeth Ellen Emily Erica Erin Esther Evelyn Felicia Felicity Flora Frances Gail Gertrude Gillian Gina Ginger Gladys Gloria Gwen Harriet Heather Helen Hilary Irene Isabel Jane Janice Jeanne Jennifer Jenny Jessica Jo Joan Joanna Joanne Jodie Josie Judith Judy Julia Julie Karen Kate Katherine Kathleen Kathryn Kathy Katie Kimberly Kirsten Kristen Kristin Laura Laurie Leah Lena Lillian Linda Lisa Liz Liza Lois Loretta Lori Lorraine Louise Lynne Marcia Margaret Maria Marian Marianne Marilyn Marjorie Marsha Mary Maureen Meg Melanie Melinda Melissa Merle Michele Michelle Miriam Molly Nan Nancy Naomi Natalie Nina Nora Norma Olivia Pam Pamela Patricia Patti Paula Pauline Peggy Phyllis Rachel Rebecca Regina Renee Rita Roberta Rosemary Sabrina Sally Samantha Sarah Selena Sheila Shelley Sherry Shirley Sonia Stacy Stephanie Sue Susanne Suzanne Suzy Sylvia Tammy Teresa Teri Terri Theresa Tina Toni Tracey Ursula Valerie Vanessa Veronica Vicki Vivian Wendy Yolanda Yvonne
almonds apple apples asparagus avocado bacon bananas barley basil bean beans beets berries berry boneless broccoli cabbage carrot carrots celery cherries cherry chile chiles chili chilies chives cilantro citrus cranberries cranberry cucumber cucumbers dill doughnuts egg eggplant eggs elk evergreen fennel figs flowers fruit fruits garlic ginger grapefruit grasses herb herbs jalapeno Jell-O lemon lemons lettuce lime lions macaroni mango maple melon mint mozzarella mushrooms oak oaks olives onion onions orange oranges orchids oregano oyster parsley pasta pastries pea peach peaches peanuts pear pears peas pecan pecans perennials pickles pine pineapple pines plum pumpkin pumpkins raspberries raspberry rice rosemary roses sage salsa scallions scallops seasonings seaweed shallots shrimp shrubs spaghetti spices spinach strawberries strawberry thyme tomato tomatoes truffles tulips turtles walnut walnuts watermelon wildflowers zucchini
mid-April mid-August mid-December mid-February mid-January mid-July mid-June mid-March mid-May mid-November mid-October mid-September mid-afternoon midafternoon midmorning midsummer

Table 4: Examples of clusters

ing 13 words and the largest cluster containing 20,396 words. Table 4 shows some examples of the generated clusters. For each cluster we list all words occurring more than 1,000 times in the corpus.

## 7 Conclusion

In this paper, we have introduced an efficient, distributed clustering algorithm for obtaining word classifications for predictive class-based language models with which we were able to use billions of tokens of training data to obtain classifications for millions of words in relatively short amounts of time.

The experiments presented show that predictive class-based models trained using the obtained word classifications can improve the quality of a state-of-the-art machine translation system as indicated by the BLEU score in both translation tasks. When using predictive class-based models in combination with a word-based language model trained on very large amounts of data, the improvements continue to be statistically significant on the *test* and *nist06* sets. We conclude that even despite the large amounts of data used to train the large word-based model in our second experiment, class-based language models are still an effective tool to ease the effects of data sparsity.

We furthermore expect to be able to increase the gains resulting from using class-based models by using more sophisticated techniques for combining them with word-based models such as linear interpolations of word- and class-based models with coefficients depending on the frequency of the history.

Another interesting direction of further research is to evaluate the use of the presented clustering technique for language model size reduction.

## References

- Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and on Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic.
- Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. 1990. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.
- Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI-04)*, San Francisco, CA, USA.
- Ahmad Emami and Frederick Jelinek. 2005. Random clusterings for language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Philadelphia, PA, USA.
- Joshua Goodman and Jianfeng Gao. 2000. Language model size reduction by pruning and clustering. In *Proceedings of the IEEE International Conference on Spoken Language Processing (ICSLP)*, Beijing, China.
- Joshua Goodman. 2000. A bit of progress in language modeling. Technical report, Microsoft Research.
- Reinherd Kneser and Hermann Ney. 1993. Improved clustering techniques for class-based statistical language modelling. In *Proceedings of the 3rd European Conference on Speech Communication and Technology*, pages 973–976, Berlin, Germany.
- Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.
- Sven Martin, Jörg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24:19–37.
- Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons, New York.
- Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, USA.
- Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA, USA.
- Martin Raab. 2006. Language model techniques in machine translation. Master’s thesis, Universität Karlsruhe / Carnegie Mellon University.
- E. W. D. Whittaker and P. C. Woodland. 2001. Efficient class-based language modelling for very large vocabularies. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 545–548, Salt Lake City, UT, USA.

# Enriching Morphologically Poor Languages for Statistical Machine Translation

**Eleftherios Avramidis**

e.avramidis@sms.ed.ac.uk

**Philipp Koehn**

pkoehn@inf.ed.ac.uk

School of Informatics  
University of Edinburgh  
2 Baccleuch Place  
Edinburgh, EH8 9LW, UK

## Abstract

We address the problem of translating from morphologically poor to morphologically rich languages by adding per-word linguistic information to the source language. We use the syntax of the source sentence to extract information for noun cases and verb persons and annotate the corresponding words accordingly. In experiments, we show improved performance for translating from English into Greek and Czech. For English–Greek, we reduce the error on the verb conjugation from 19% to 5.4% and noun case agreement from 9% to 6%.

## 1 Introduction

Traditional statistical machine translation methods are based on mapping on the lexical level, which takes place in a local window of a few words. Hence, they fail to produce adequate output in many cases where more complex linguistic phenomena play a role. Take the example of morphology. Predicting the correct morphological variant for a target word may not depend solely on the source words, but require additional information about its role in the sentence.

Recent research on handling rich morphology has largely focused on translating from rich morphology languages, such as Arabic, into English (Habash and Sadat, 2006). There has been less work on the opposite case, translating from English into morphologically richer languages. In a study of translation quality for languages in the Europarl corpus, Koehn (2005) reports that translating into morphologically

richer languages is more difficult than translating from them.

There are intuitive reasons why generating richer morphology from morphologically poor languages is harder. Take the example of translating noun phrases from English to Greek (or German, Czech, etc.). In English, a noun phrase is rendered the same if it is the subject or the object. However, Greek words in noun phrases are inflected based on their role in the sentence. A purely lexical mapping of English noun phrases to Greek noun phrases suffers from the lack of information about its role in the sentence, making it hard to choose the right inflected forms.

Our method is based on factored phrase-based statistical machine translation models. We focused on preprocessing the source data to acquire the needed information and then use it within the models. We mainly carried out experiments on English to Greek translation, a language pair that exemplifies the problems of translating from a morphologically poor to a morphologically rich language.

### 1.1 Morphology in Phrase-based SMT

When examining parallel sentences of such language pairs, it is apparent that for many English words and phrases which appear usually in the same form, the corresponding terms of the richer target language appear inflected in many different ways. On a single word-based probabilistic level, it is then obvious that for one specific English word  $e$  the probability  $p(f|e)$  of it being translated into a word  $f$  decreases as the number of translation candidates increase, making the decisions more uncertain.

- **English:** The president, after reading the press review and the announcements, left his office
- **Greek-1:** The president[nominative], after reading[3<sup>rd</sup>sing] the press review[accusative,sing] and the announcements[accusative,plur], left[3<sup>rd</sup>sing] his office[accusative,sing]
- **Greek-2:** The president[nominative], after reading[3<sup>rd</sup>sing] the press review[accusative,sing] and the announcements[nominative,plur], left[3<sup>rd</sup>plur] his office[accusative,sing]

Figure 1: Example of missing agreement information, affecting the meaning of the second sentence

One of the main aspects required for the fluency of a sentence is *agreement*. Certain words have to match in gender, case, number, person etc. within a sentence. The exact rules of agreement are language-dependent and are closely linked to the morphological structure of the language.

Traditional statistical machine translation models deal with this problems in two ways:

- The basic SMT approach uses the target language model as a feature in the argument maximisation function. This language model is trained on grammatically correct text, and would therefore give a good probability for word sequences that are likely to occur in a sentence, while it would penalise ungrammatical or badly ordered formations.
- Meanwhile, in phrase-based SMT models, words are mapped in chunks. This can resolve phenomena where the English side uses more than one words to describe what is denoted on the target side by one morphologically inflected term.

Thus, with respect to these methods, there is a problem when agreement needs to be applied on part of a sentence whose length exceeds the order of the of the target  $n$ -gram language model and the size of the chunks that are translated (see Figure 1 for an example).

## 1.2 Related Work

In one of the first efforts to enrich the source in word-based SMT, Ueffing and Ney (2003) used part-of-speech (POS) tags, in order to deal with the verb conjugation of Spanish and Catalan; so, POS tags were used to identify the pronoun+verb sequence and splice these two words into one term. The approach was clearly motivated by the problems occurring by a single-word-based SMT and have been solved by adopting a phrase-based model. Meanwhile, there is no handling of the case when the pronoun stays in distance with the related verb.

Minkov et al. (2007) suggested a post-processing system which uses morphological and syntactic features, in order to ensure grammatical agreement on the output. The method, using various grammatical source-side features, achieved higher accuracy when applied directly to the reference translations but it was not tested as a part of an MT system. Similarly, translating English into Turkish (Durgar El-Kahlout and Oflazer, 2006) uses POS and morph stems in the input along with rich Turkish morph tags on the target side, but improvement was gained only after augmenting the generation process with morphotactical knowledge. Habash et al. (2007) also investigated case determination in Arabic. Carpuat and Wu (2007) approached the issue as a Word Sense Disambiguation problem.

In their presentation of the factored SMT models, Koehn and Hoang (2007) describe experiments for translating from English to German, Spanish and Czech, using morphology tags added on the morphologically rich side, along with POS tags. The morphological factors are added on the morphologically rich side and scored with a 7-gram sequence model. Probabilistic models for using only source tags were investigated by Birch et al. (2007), who attached syntax hints in factored SMT models by having *Combinatorial Categorical Grammar (CCG) supertags* as factors on the input words, but in this case English was the target language.

This paper reports work that strictly focuses on translation from English to a morphologically richer language. We go one step further than just using easily acquired information (e.g. English POS or lemmata) and extract target-specific information from the source sentence context. We use syntax, not in



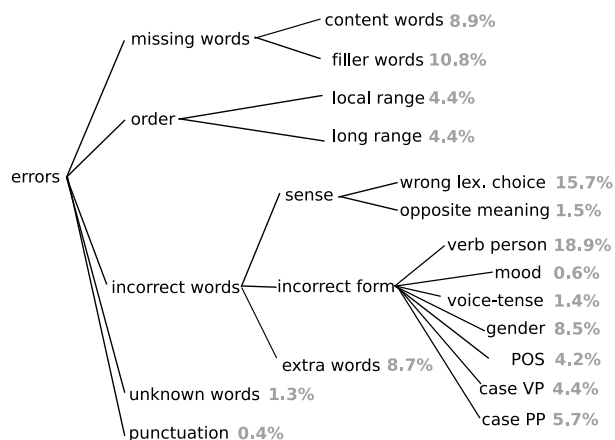


Figure 2: Classification of the errors on our English-Greek baseline system (ch. 4.1), as suggested by Vilar et al. (2006)

order to aid reordering (Yamada and Knight, 2001; Collins et al., 2005; Huang et al., 2006), but as a means for getting the “missing” morphology information, depending on the syntactic position of the words of interest. Then, contrary to the methods that added only output features or altered the generation procedure, we used this information in order to augment only the source side of a factored translation model, assuming that we do not have resources allowing factors or specialized generation in the target language (a common problem, when translating from English into under-resourced languages).

## 2 Methods for enriching input

We selected to focus on *noun cases agreement* and *verb person conjugation*, since they were the most frequent grammatical errors of our baseline SMT system (see full error analysis in Figure 2). Moreover, these types of inflection signify the constituents of every phrase, tightly linked to the meaning of the sentence.

### 2.1 Case agreement

The case agreement for nouns, adjectives and articles is mainly defined by the syntactic role that each noun phrase has. Nominative case is used to define the nouns which are the subject of the sentence, accusative shows usually the direct object of the verbs and dative case refers to the indirect object of bi-transitive verbs.

Therefore, the followed approach takes advantage of syntax, following a method similar to *Semantic Role Labelling* (Carreras and Marquez, 2005; Surdeanu and Turmo, 2005). English, as morphologically poor language, usually follows a fixed word order (subject-verb-object), so that a syntax parser can be easily used for identifying the subject and the object of most sentences. Considering such annotation, a factored translation model is trained to map the word-case pair to the correct inflection of the target noun. Given the agreement restriction, all words that accompany the noun (adjectives, articles, determiners) must follow the case of the noun, so their likely case needs to be identified as well.

For this purpose we use a syntax parser to acquire the syntax tree for each English sentence. The trees are parsed depth-first and the cases are identified within particular “sub-tree patterns” which are manually specified. We use the sequence of the nodes in the tree to identify the syntactic role of each noun phrase.

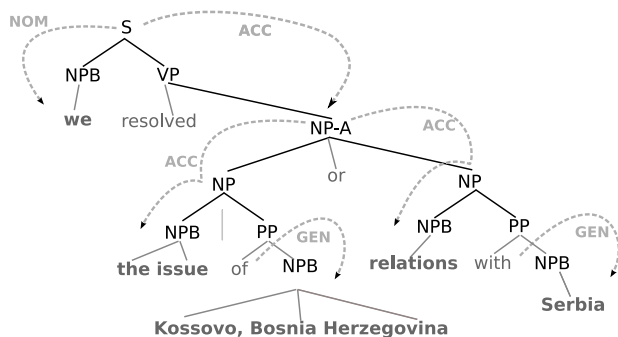


Figure 3: Case tags are assigned on depth-first parse of the English syntax tree, based on sub-tree patterns

To make things more clear, an example can be seen in figure 3. At first, the algorithm identifies the subtree “S-(NPB-VP)” and the *nominative* tag is applied on the NPB node, so that it is assigned to the word “we” (since a pronoun can have a case). The example of accusative shows how cases get transferred to nested subtrees. In practice, they are recursively transferred to every underlying noun phrase (NP) but not to clauses that do not need this information (e.g. prepositional phrases). Similar rules are applied for covering a wide range of node sequence patterns.

Also note that this method had to be target-

oriented in some sense: we considered the target language rules for choosing the noun case in every prepositional phrase, depending on the leading preposition. This way, almost all nouns were tagged and therefore the number of the factored words was increased, in an effort to decrease sparsity. Similarly, cases which do not actively affect morphology (e.g. dative in Greek) were not tagged during factorization.

## 2.2 Verb person conjugation

For resolving the verb conjugation, we needed to identify the person of a verb and add this piece of linguistic information as a tag. As we parse the tree top-down, on every level, we look for two discrete nodes which, somewhere in their children, include the verb and the corresponding subject. Consequently, the node which contains the subject is searched recursively until a subject is found. Then, the person is identified and the tag is assigned to the node which contains the verb, which recursively bequeaths this tag to the nested subtree.

For the subject selection, the following rules were applied:

- The verb person is directly connected to the subject of the sentence and in most cases it is directly inferred by a personal pronoun (I, you etc). Therefore, since this is usually the case, when a pronoun existed, it was directly used as a tag.
- All pronouns in a different case (e.g. *them*, *myself*) were converted into nominative case before being used as a tag.
- When the subject of the sentence is not a pronoun, but a single noun, then it is in third person. The POS tag of this noun is then used to identify if it is plural or singular. This was selectively modified for nouns which despite being in singular, take a verb in plural.
- The gender of the subject does not affect the inflection of the verb in Greek. Therefore, all three genders that are given by the third person pronouns were reduced to one.

In Figure 4 we can see an example of how the person tag is extracted from the subject of the sen-

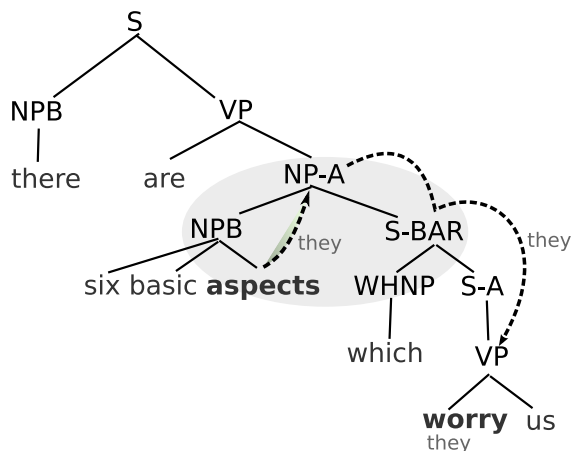


Figure 4: Applying person tags on an English syntax tree

tence and gets passed to the relative clause. In particular, as the algorithm parses the syntax tree, it identifies the sub-tree which has NP-A as a head and includes the WHNP node. Consequently, it recursively browses the preceding NPB so as to get the subject of the sentence. The word “aspects” is found, which has a POS tag that shows it is a plural noun. Therefore, we consider the subject to be of the third person in plural (tagged by *they*) which is recursively passed to the children of the head node.

## 3 Factored Model

The factored statistical machine translation model uses a log-linear approach, in order to combine the several components, including the language model, the reordering model, the translation models and the generation models. The model is defined mathematically (Koehn and Hoang, 2007) as following:

$$p(\mathbf{f}|\mathbf{e}) = \frac{1}{Z} \exp \sum_{i=1}^n \lambda_i h_i(\mathbf{f}, \mathbf{e}) \quad (1)$$

where  $\lambda_i$  is a vector of weights determined during a tuning process, and  $h_i$  is the feature function. The feature function for a translation probability distribution is

$$h_T(\mathbf{f}|\mathbf{e}) = \sum_j \tau(\bar{e}_j, \bar{f}_j) \quad (2)$$

While factored models may use a generation step to combine the several translation components based on the output factors, we use only source factors;

therefore we don't need a generation step to combine the probabilities of the several components.

Instead, factors are added so that both words and its factor(s) are assigned the same probability. Of course, when there is not 1-1 mapping between the *word+factor* splice on the source and the inflected word on the target, the well-known issue of sparse data arises. In order to reduce these problems, decoding needed to consider alternative paths to translation tables trained with less or no factors (as Birch et al. (2007) suggested), so as to cover instances where a word appears with a factor which it has not been trained with. This is similar to back-off. The alternative paths are combined as following (fig. 5):

$$h_T(\mathbf{f}|\mathbf{e}) = \sum_j h_{T_{t(j)}}(\bar{e}_j, \bar{f}_j) \quad (3)$$

where each phrase  $j$  is translated by one translation table  $t(j)$  and each table  $i$  has a feature function  $h_{T_i}$ , as shown in eq. (2).

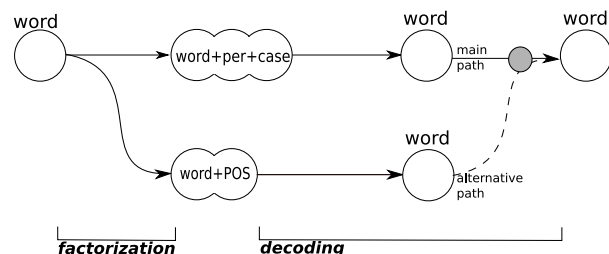


Figure 5: Decoding using an alternative path with different factorization

## 4 Experiments

This preprocessing led to annotated source data, which were given as an input to a factored SMT system.

### 4.1 Experiment setup

For testing the factored translation systems, we used Moses (Koehn et al., 2007), along with a 5-gram SRILM language model (Stolcke, 2002). A Greek model was trained on 440,082 aligned sentences of Europarl v.3, tuned with *Minimum Error Training* (Och, 2003). It was tuned over a development set of 2,000 Europarl sentences and tested on two sets of 2,000 sentences each, from the Europarl and a

News Commentary respectively, following the specifications made by the ACL 2007 2nd Workshop on SMT<sup>1</sup>. A Czech model was trained on 57,464 aligned sentences, tuned over 1057 sentences of the News Commentary corpus and tested on two sets of 964 sentences and 2000 sentences respectively.

The training sentences were trimmed to a length of 60 words for reducing perplexity and a standard lexicalised reordering, with distortion limit set to 6. For getting the syntax trees, the latest version of Collins' parser (Collins, 1997) was used. When needed, part-of-speech (POS) tags were acquired by using Brill's tagger (Brill, 1992) on v1.14. Results were evaluated with both BLEU (Papineni et al., 2001) and NIST metrics (NIST, 2002).

### 4.2 Results

set	BLEU		NIST	
	devtest	test07	devtest	test07
baseline	18.13	18.05	5.218	5.279
person	18.16	18.17	5.224	5.316
pos+person	18.14	18.16	5.259	5.316
person+case	18.08	18.24	5.258	5.340
<i>altpath</i> :POS	18.21	18.20	5.285	5.340

Table 1: Translating English to Greek: Using a single translation table may cause sparse data problems, which are addressed using an alternative path to a second translation table

We tested several various combinations of tags, while using a single translation component. Some combinations seem to be affected by sparse data problems and the best score is achieved by using both person and case tags. Our full method, using both factors, was more effective on the second test-set, but the best score in average was succeeded by using an alternative path to a POS-factored translation table (table 1). The NIST metric clearly shows a significant improvement, because it mostly measures difficult n-gram matches (e.g. due to the long-distance rules we have been dealing with).

<sup>1</sup>see <http://www.statmt.org/wmt07> referring to sets *dev2006* (tuning) and *devtest2006*, *test2007* (testing)

### 4.3 Error analysis

In  $n$ -gram based metrics, the scores for all words are equally weighted, so mistakes on crucial sentence constituents may be penalized the same as errors on redundant or meaningless words (Callison-Burch et al., 2006). We consider agreement on verbs and nouns an important factor for the adequacy of the result, since they adhere more to the semantics of the sentence. Since we targeted these problems, we conducted a manual error analysis focused on the success of the improved system regarding those specific phenomena.

system	verbs	errors	missing
baseline	311	19.0%	7.4%
single	295	4.7%	5.4%
alt.path	294	5.4%	2.7%

Table 2: Error analysis of 100 test sentences, focused on verb person conjugation, for using both person and case tags

system	NPs	errors	missing
baseline	469	9.0%	4.9%
single	465	6.2%	4.5%
alt. path	452	6.0%	4.0%

Table 3: Error analysis of 100 test sentences, focused on noun cases, for using both person and case tags

The analysis shows that using a system with only one phrase translation table caused a high percentage of missing or untranslated words. When a word appears with a tag with which it has not been trained, that would be considered an unseen event and remain untranslated. The use of the alternative path seems to be a good solution.

step	parsing	tagging	decoding
VPs	16.7%	25%	58.3%
NPs	39.2%	21.7%	39.1%
avg	31.4%	22.9%	45.7 %

Table 4: Analysis on which step of the translation process the agreement errors derive from, based on manual resolution on the errors of table 3

The impact of the preprocessing stage to the errors may be seen in table 4, where errors are tracked

back to the stage they derived from. Apart from the decoding errors, which may be attributed to sparse data or other statistical factors, a large part of the errors derive from the preprocessing step; either the syntax tree of the sentence was incorrectly or partially resolved, or our labelling process did not correctly match all possible sub-trees.

### 4.4 Investigating applicability to other inflected languages

The grammatical phenomena of noun cases and verb persons are quite common among many human languages. While the method was tested in Greek, there was an effort to investigate whether it is useful for other languages with similar characteristics. For this reason, the method was adapted for Czech, which needs agreement on both verb conjugation and 9 noun cases. Dative case was included for the indirect object and the rules of the prepositional phrases were adapted to tag all three cases that can be verb phrase constituents. The Czech noun cases which appear only in prepositional phrases were ignored, since they are covered by the phrase-based model.

set	BLUE		NIST	
	devtest	test	devtest	test
baseline	12.08	12.34	4.634	4.865
person+case <i>altpath</i> :POS	11.98	11.99	4.584	4.801
person <i>altpath</i> :word	12.23	12.11	4.647	4.846
case <i>altpath</i> :word	12.54	12.51	4.758	4.957

Table 5: Enriching source data can be useful when translating from English to Czech, since it is a morphologically rich language. Experiments shown improvement when using factors on noun-cases with an alternative path

In Czech, due to the small size of the corpus, it was possible to improve metric scores only by using an alternative path to a bare word-to-word translation table. Combining case and verb tags worsened the results, which suggests that, while applying the method to more languages, a different use of the attributes may be beneficial for each of them.

## 5 Conclusion

In this paper we have shown how SMT performance can be improved, when translating from English into morphologically richer languages, by adding linguistic information on the source. Although the source language misses morphology attributes required by the target language, the needed information is inherent in the syntactic structure of the source sentence. Therefore, we have shown that this information can be easily be included in a SMT model by preprocessing the source text.

Our method focuses on two linguistic phenomena which produce common errors on the output and are important constituents of the sentence. In particular, noun cases and verb persons are required by the target language, but not directly inferred by the source. For each of the sub-problems, our algorithm used heuristic syntax-based rules on the statistically generated syntax tree of each sentence, in order to address the missing information, which was consequently tagged in by means of word factors. This information was proven to improve the outcome of a factored SMT model, by reducing the grammatical agreement errors on the generated sentences.

An initial system using one translation table with additional source side factors caused sparse data problems, due to the increased number of unseen word-factor combinations. Therefore, the decoding process is given an *alternative path* towards a translation table with less or no factors.

The method was tested on translating from English into two morphologically rich languages. Note that this may be easily expanded for translating from English into many morphologically richer languages with similar attributes. Opposed to other factored translation model approaches that require target language factors, that are not easily obtainable for many languages, our approach only requires English syntax trees, which are acquired with widely available automatic parsers. The preprocessing scripts were adapted so that they provide the morphology attributes required by the target language and the best combination of factors and alternative paths was chosen.

## Acknowledgments

This work was supported in part under the EuroMatrix project funded by the European Commission (6th Framework Programme). Many thanks to Josh Schroeder for preparing the training, development and test data for Greek, in accordance to the standards of ACL 2007 2nd Workshop on SMT; to Hieu Hoang, Alexandra Birch and all the members of the Edinburgh University SMT group for answering questions, making suggestions and providing support.

## References

- Birch, A., Osborne, M., and Koehn, P. 2007. CCG Supertags in factored Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16, Prague, Czech Republic. Association for Computational Linguistics.
- Brill, E. 1992. A simple rule-based part of speech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 152–155.
- Callison-Burch, C., Osborne, M., and Koehn, P. 2006. Re-evaluation the role of bleu in machine translation research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*. The Association for Computer Linguistics.
- Carpuat, M. and Wu, D. 2007. Improving Statistical Machine Translation using Word Sense Disambiguation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 61–72, Prague, Czech Republic.
- Carreras, X. and Marquez, L. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of 9th Conference on Computational Natural Language Learning (CoNLL)*, pages 169–172, Ann Arbor, Michigan, USA.
- Collins, M. 1997. Three generative, lexicalised models for statistical parsing. *Proceedings of the 35th conference on Association for Computational Linguistics*, pages 16–23.
- Collins, M., Koehn, P., and Kučerová, I. 2005. Clause restructuring for statistical machine translation. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 531–540, Morristown, NJ, USA. Association for Computational Linguistics.

- Durgar El-Kahlout, i. and Oflazer, K. 2006. Initial explorations in english to turkish statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 7–14, New York City. Association for Computational Linguistics.
- Habash, N., Gabbard, R., Rambow, O., Kulick, S., and Marcus, M. 2007. Determining case in Arabic: Learning complex linguistic behavior requires complex linguistic features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1084–1092.
- Habash, N. and Sadat, F. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 49–52, New York City, USA. Association for Computational Linguistics.
- Huang, L., Knight, K., and Joshi, A. 2006. Statistical syntax-directed translation with extended domain of locality. *Proc. AMTA*, pages 66–73.
- Koehn, P. 2005. Europarl: A parallel corpus for statistical machine translation. *MT Summit*, 5.
- Koehn, P. and Hoang, H. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876.
- Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.
- Minkov, E., Toutanova, K., and Suzuki, H. 2007. Generating complex morphology for machine translation. In *ACL 07: Proceedings of the 45th Annual Meeting of the Association of Computational linguistics*, pages 128–135, Prague, Czech Republic. Association for Computational Linguistics.
- NIST 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics.
- Och, F. J. 2003. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.
- Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. 2001. BLEU: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.
- Stolcke, A. 2002. SRILM-an extensible language modeling toolkit. *Proc. ICSLP*, 2:901–904.
- Surdeanu, M. and Turmo, J. 2005. Semantic Role Labeling Using Complete Syntactic Analysis. In *Proceedings of 9th Conference on Computational Natural Language Learning (CoNLL)*, pages 221–224, Ann Arbor, Michigan, USA.
- Ueffing, N. and Ney, H. 2003. Using pos information for statistical machine translation into morphologically rich languages. In *EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, pages 347–354, Morristown, NJ, USA. Association for Computational Linguistics.
- Vilar, D., Xu, J., D'Haro, L. F., and Ney, H. 2006. Error Analysis of Machine Translation Output. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'06)*, pages 697–702, Genoa, Italy.
- Yamada, K. and Knight, K. 2001. A syntax-based statistical translation model. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530, Morristown, NJ, USA. Association for Computational Linguistics.

# Learning Bilingual Lexicons from Monolingual Corpora

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick and Dan Klein

Computer Science Division, University of California at Berkeley

{aria42, pliang, tberg, klein}@cs.berkeley.edu

## Abstract

We present a method for learning bilingual translation lexicons from monolingual corpora. Word types in each language are characterized by purely monolingual features, such as context counts and orthographic substrings. Translations are induced using a generative model based on canonical correlation analysis, which explains the monolingual lexicons in terms of latent matchings. We show that high-precision lexicons can be learned in a variety of language pairs and from a range of corpus types.

## 1 Introduction

Current statistical machine translation systems use parallel corpora to induce translation correspondences, whether those correspondences be at the level of phrases (Koehn, 2004), treelets (Galley et al., 2006), or simply single words (Brown et al., 1994). Although parallel text is plentiful for some language pairs such as English-Chinese or English-Arabic, it is scarce or even non-existent for most others, such as English-Hindi or French-Japanese. Moreover, parallel text could be scarce for a language pair even if monolingual data is readily available for both languages.

In this paper, we consider the problem of learning translations from monolingual sources alone. This task, though clearly more difficult than the standard parallel text approach, can operate on language pairs and in domains where standard approaches cannot. We take as input two monolingual corpora and perhaps some seed translations, and we produce as output a *bilingual lexicon*, defined as a list of word

pairs deemed to be word-level translations. Precision and recall are then measured over these bilingual lexicons. This setting has been considered before, most notably in Koehn and Knight (2002) and Fung (1995), but the current paper is the first to use a probabilistic model and present results across a variety of language pairs and data conditions.

In our method, we represent each language as a *monolingual lexicon* (see figure 2): a list of word types characterized by monolingual feature vectors, such as context counts, orthographic substrings, and so on (section 5). We define a generative model over (1) a source lexicon, (2) a target lexicon, and (3) a matching between them (section 2). Our model is based on *canonical correlation analysis* (CCA)<sup>1</sup> and explains matched word pairs via vectors in a common latent space. Inference in the model is done using an EM-style algorithm (section 3).

Somewhat surprisingly, we show that it is possible to learn or extend a translation lexicon using monolingual corpora alone, in a variety of languages and using a variety of corpora, even in the absence of orthographic features. As might be expected, the task is harder when no seed lexicon is provided, when the languages are strongly divergent, or when the monolingual corpora are from different domains. Nonetheless, even in the more difficult cases, a sizable set of high-precision translations can be extracted. As an example of the performance of the system, in English-Spanish induction with our best feature set, using corpora derived from topically similar but non-parallel sources, the system obtains 89.0% precision at 33% recall.

<sup>1</sup>See Hardoon et al. (2003) for an overview.

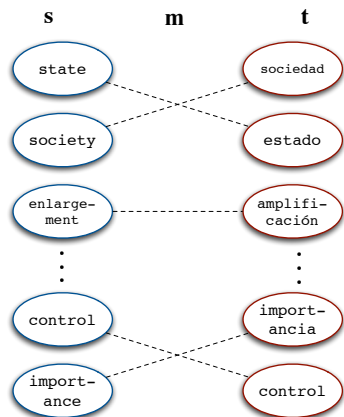


Figure 1: Bilingual lexicon induction: source word types  $s$  are listed on the left and target word types  $t$  on the right. Dashed lines between nodes indicate translation pairs which are in the matching  $\mathbf{m}$ .

## 2 Bilingual Lexicon Induction

As input, we are given a monolingual corpus  $S$  (a sequence of word tokens) in a *source* language and a monolingual corpus  $T$  in a *target* language. Let  $\mathbf{s} = (s_1, \dots, s_{n_S})$  denote  $n_S$  word types appearing in the source language, and  $\mathbf{t} = (t_1, \dots, t_{n_T})$  denote word types in the target language. Based on  $S$  and  $T$ , our goal is to output a matching  $\mathbf{m}$  between  $\mathbf{s}$  and  $\mathbf{t}$ . We represent  $\mathbf{m}$  as a set of integer pairs so that  $(i, j) \in \mathbf{m}$  if and only if  $s_i$  is matched with  $t_j$ .

### 2.1 Generative Model

We propose the following generative model over matchings  $\mathbf{m}$  and word types  $(\mathbf{s}, \mathbf{t})$ , which we call *matching canonical correlation analysis (MCCA)*.

MCCA model	
$\mathbf{m} \sim \text{MATCHING-PRIOR}$	[matching $\mathbf{m}$ ]
For each matched edge $(i, j) \in \mathbf{m}$ :	
$z_{i,j} \sim \mathcal{N}(0, I_d)$	[latent concept]
$f_S(s_i) \sim \mathcal{N}(W_S z_{i,j}, \Psi_S)$	[source features]
$f_T(t_j) \sim \mathcal{N}(W_T z_{i,j}, \Psi_T)$	[target features]
For each unmatched source word type $i$ :	
$f_S(s_i) \sim \mathcal{N}(0, \sigma^2 I_{d_S})$	[source features]
For each unmatched target word type $j$ :	
$f_T(t_j) \sim \mathcal{N}(0, \sigma^2 I_{d_T})$	[target features]

First, we generate a matching  $\mathbf{m} \in \mathcal{M}$ , where  $\mathcal{M}$  is the set of matchings in which each word type is

matched to at most one other word type.<sup>2</sup> We take MATCHING-PRIOR to be uniform over  $\mathcal{M}$ .<sup>3</sup>

Then, for each matched pair of word types  $(i, j) \in \mathbf{m}$ , we need to generate the observed feature vectors of the source and target word types,  $f_S(s_i) \in \mathbb{R}^{d_S}$  and  $f_T(t_j) \in \mathbb{R}^{d_T}$ . The feature vector of each word type is computed from the appropriate monolingual corpus and summarizes the word’s monolingual characteristics; see section 5 for details and figure 2 for an illustration. Since  $s_i$  and  $t_j$  are translations of each other, we expect  $f_S(s_i)$  and  $f_T(t_j)$  to be connected somehow by the generative process. In our model, they are related through a vector  $z_{i,j} \in \mathbb{R}^d$  representing the shared, language-independent concept.

Specifically, to generate the feature vectors, we first generate a random concept  $z_{i,j} \sim \mathcal{N}(0, I_d)$ , where  $I_d$  is the  $d \times d$  identity matrix. The source feature vector  $f_S(s_i)$  is drawn from a multivariate Gaussian with mean  $W_S z_{i,j}$  and covariance  $\Psi_S$ , where  $W_S$  is a  $d_S \times d$  matrix which transforms the language-independent concept  $z_{i,j}$  into a language-dependent vector in the source space. The arbitrary covariance parameter  $\Psi_S \succeq 0$  explains the source-specific variations which are not captured by  $W_S$ ; it does not play an explicit role in inference. The target  $f_T(t_j)$  is generated analogously using  $W_T$  and  $\Psi_T$ , conditionally independent of the source given  $z_{i,j}$  (see figure 2). For each of the remaining unmatched source word types  $s_i$  which have not yet been generated, we draw the word type features from a baseline normal distribution with variance  $\sigma^2 I_{d_S}$ , with hyperparameter  $\sigma^2 \gg 0$ ; unmatched target words are similarly generated.

If two word types are truly translations, it will be better to relate their feature vectors through the latent space than to explain them independently via the baseline distribution. However, if a source word type is not a translation of any of the target word types, we can just generate it independently without requiring it to participate in the matching.

<sup>2</sup>Our choice of  $\mathcal{M}$  permits unmatched word types, but does not allow words to have multiple translations. This setting facilitates comparison to previous work and admits simpler models.

<sup>3</sup>However, non-uniform priors could encode useful information, such as rank similarities.



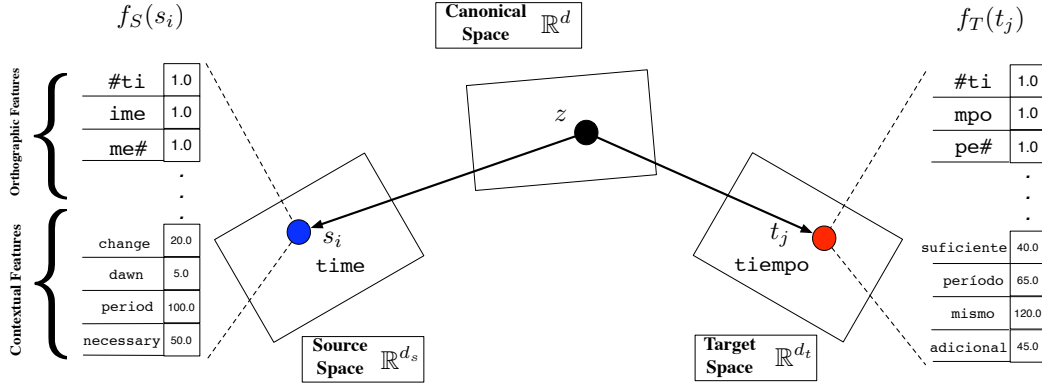


Figure 2: Illustration of our MCCA model. Each latent concept  $z_{i,j}$  originates in the canonical space. The observed word vectors in the source and target spaces are generated independently given this concept.

### 3 Inference

Given our probabilistic model, we would like to maximize the log-likelihood of the observed data  $(\mathbf{s}, \mathbf{t})$ :

$$\ell(\theta) = \log p(\mathbf{s}, \mathbf{t}; \theta) = \log \sum_{\mathbf{m}} p(\mathbf{m}, \mathbf{s}, \mathbf{t}; \theta)$$

with respect to the model parameters  $\theta = (W_S, W_T, \Psi_S, \Psi_T)$ .

We use the hard (Viterbi) EM algorithm as a starting point, but due to modeling and computational considerations, we make several important modifications, which we describe later. The general form of our algorithm is as follows:

Summary of learning algorithm

**E-step:** Find the maximum weighted (partial) bipartite matching  $\mathbf{m} \in \mathcal{M}$

**M-step:** Find the best parameters  $\theta$  by performing canonical correlation analysis (CCA)

**M-step** Given a matching  $\mathbf{m}$ , the M-step optimizes  $\log p(\mathbf{m}, \mathbf{s}, \mathbf{t}; \theta)$  with respect to  $\theta$ , which can be rewritten as

$$\max_{\theta} \sum_{(i,j) \in \mathbf{m}} \log p(s_i, t_j; \theta). \quad (1)$$

This objective corresponds exactly to maximizing the likelihood of the probabilistic CCA model presented in Bach and Jordan (2006), which proved that the maximum likelihood estimate can be computed by *canonical correlation analysis* (CCA). Intuitively, CCA finds  $d$ -dimensional subspaces  $U_S \in$

$\mathbb{R}^{d_S \times d}$  of the source and  $U_T \in \mathbb{R}^{d_T \times d}$  of the target such that the components of the projections  $U_S^\top f_S(s_i)$  and  $U_T^\top f_T(t_j)$  are maximally correlated.<sup>4</sup>

$U_S$  and  $U_T$  can be found by solving an eigenvalue problem (see Haroon et al. (2003) for details). Then the maximum likelihood estimates are as follows:  $W_S = C_{SS} U_S P^{1/2}$ ,  $W_T = C_{TT} U_T P^{1/2}$ ,  $\Psi_S = C_{SS} - W_S W_S^\top$ , and  $\Psi_T = C_{TT} - W_T W_T^\top$ , where  $P$  is a  $d \times d$  diagonal matrix of the canonical correlations,  $C_{SS} = \frac{1}{|\mathbf{m}|} \sum_{(i,j) \in \mathbf{m}} f_S(s_i) f_S(s_i)^\top$  is the empirical covariance matrix in the source domain, and  $C_{TT}$  is defined analogously.

**E-step** To perform a conventional E-step, we would need to compute the posterior over all matchings, which is #P-complete (Valiant, 1979). On the other hand, hard EM only requires us to compute the best matching under the current model:<sup>5</sup>

$$\mathbf{m} = \operatorname{argmax}_{\mathbf{m}'} \log p(\mathbf{m}', \mathbf{s}, \mathbf{t}; \theta). \quad (2)$$

We cast this optimization as a maximum weighted bipartite matching problem as follows. Define the edge weight between source word type  $i$  and target word type  $j$  to be

$$w_{i,j} = \log p(s_i, t_j; \theta) - \log p(s_i; \theta) - \log p(t_j; \theta), \quad (3)$$

<sup>4</sup>Since  $d_S$  and  $d_T$  can be quite large in practice and often greater than  $|\mathbf{m}|$ , we use Cholesky decomposition to represent the feature vectors as  $|\mathbf{m}|$ -dimensional vectors with the same dot products, which is all that CCA depends on.

<sup>5</sup>If we wanted softer estimates, we could use the agreement-based learning framework of Liang et al. (2008) to combine two tractable models.

which can be loosely viewed as a pointwise mutual information quantity. We can check that the objective  $\log p(\mathbf{m}, \mathbf{s}, \mathbf{t}; \theta)$  is equal to the weight of a matching plus some constant  $C$ :

$$\log p(\mathbf{m}, \mathbf{s}, \mathbf{t}; \theta) = \sum_{(i,j) \in \mathbf{m}} w_{i,j} + C. \quad (4)$$

To find the optimal partial matching, edges with weight  $w_{i,j} < 0$  are set to zero in the graph and the optimal full matching is computed in  $O((n_S + n_T)^3)$  time using the Hungarian algorithm (Kuhn, 1955). If a zero edge is present in the solution, we remove the involved word types from the matching.<sup>6</sup>

**Bootstrapping** Recall that the E-step produces a partial matching of the word types. If too few word types are matched, learning will not progress quickly; if too many are matched, the model will be swamped with noise. We found that it was helpful to explicitly control the number of edges. Thus, we adopt a bootstrapping-style approach that only permits high confidence edges at first, and then slowly permits more over time. In particular, we compute the optimal full matching, but only retain the highest weighted edges. As we run EM, we gradually increase the number of edges to retain.

In our context, bootstrapping has a similar motivation to the annealing approach of Smith and Eisner (2006), which also tries to alter the space of hidden outputs in the E-step over time to facilitate learning in the M-step, though of course the use of bootstrapping in general is quite widespread (Yarowsky, 1995).

## 4 Experimental Setup

In section 5, we present developmental experiments in English-Spanish lexicon induction; experiments

<sup>6</sup>Empirically, we obtained much better efficiency and even increased accuracy by replacing these marginal likelihood weights with a simple proxy, the distances between the words’ mean latent concepts:

$$w_{i,j} = A - \|z_i^* - z_j^*\|_2, \quad (5)$$

where  $A$  is a thresholding constant,  $z_i^* = \mathbb{E}(z_{i,j} \mid f_S(s_i)) = P^{1/2} U_S^\top f_S(s_i)$ , and  $z_j^*$  is defined analogously. The increased accuracy may not be an accident: whether two words are translations is perhaps better characterized directly by how close their latent concepts are, whereas log-probability is more sensitive to perturbations in the source and target spaces.

are presented for other languages in section 6. In this section, we describe the data and experimental methodology used throughout this work.

### 4.1 Data

Each experiment requires a source and target monolingual corpus. We use the following corpora:

- EN-ES-W: 3,851 Wikipedia articles with both English and Spanish bodies (generally not direct translations).
- EN-ES-P: 1st 100k sentences of text from the parallel English and Spanish Europarl corpus (Koehn, 2005).
- EN-ES(FR)-D: English: 1st 50k sentences of Europarl; Spanish (French): 2nd 50k sentences of Europarl.<sup>7</sup>
- EN-CH-D: English: 1st 50k sentences of Xinhua parallel news corpora;<sup>8</sup> Chinese: 2nd 50k sentences.
- EN-AR-D: English: 1st 50k sentences of 1994 proceedings of UN parallel corpora;<sup>9</sup> Arabic: 2nd 50k sentences.
- EN-ES-G: English: 100k sentences of English Gigaword; Spanish: 100k sentences of Spanish Gigaword.<sup>10</sup>

Note that even when corpora are derived from parallel sources, no explicit use is ever made of document or sentence-level alignments. In particular, our method is robust to permutations of the sentences in the corpora.

### 4.2 Lexicon

Each experiment requires a lexicon for evaluation. Following Koehn and Knight (2002), we consider lexicons over only noun word types, although this is not a fundamental limitation of our model. We consider a word type to be a noun if its most common tag is a noun in our monolingual corpus.<sup>11</sup> For

<sup>7</sup>Note that although the corpora here are derived from a parallel corpus, there are no parallel sentences.

<sup>8</sup>LDC catalog # 2002E18.

<sup>9</sup>LDC catalog # 2004E13.

<sup>10</sup>These corpora contain no parallel sentences.

<sup>11</sup>We use the Tree Tagger (Schmid, 1994) for all POS tagging except for Arabic, where we use the tagger described in Diab et al. (2004).

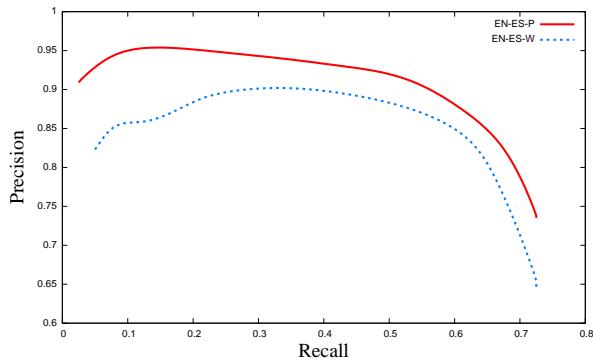


Figure 3: Example precision/recall curve of our system on EN-ES-P and EN-ES-W settings. See section 6.1.

all languages pairs except English-Arabic, we extract evaluation lexicons from the *Wiktionary* online dictionary. As we discuss in section 7, our extracted lexicons have low coverage, particularly for proper nouns, and thus all performance measures are (sometimes substantially) pessimistic. For English-Arabic, we extract a lexicon from 100k parallel sentences of UN parallel corpora by running the HMM intersected alignment model (Liang et al., 2008), adding  $(s, t)$  to the lexicon if  $s$  was aligned to  $t$  at least three times and more than any other word.

Also, as in Koehn and Knight (2002), we make use of a *seed lexicon*, which consists of a small, and perhaps incorrect, set of initial translation pairs. We used two methods to derive a seed lexicon. The first is to use the evaluation lexicon  $\mathcal{L}_e$  and select the hundred most common noun word types in the source corpus which have translations in  $\mathcal{L}_e$ . The second method is to heuristically induce, where applicable, a seed lexicon using edit distance, as is done in Koehn and Knight (2002). Section 6.2 compares the performance of these two methods.

### 4.3 Evaluation

We evaluate a proposed lexicon  $\mathcal{L}_p$  against the evaluation lexicon  $\mathcal{L}_e$  using the  $F_1$  measure in the standard fashion; precision is given by the number of proposed translations contained in the evaluation lexicon, and recall is given by the fraction of possible translation pairs proposed.<sup>12</sup> Since our model

<sup>12</sup>We should note that precision is not penalized for  $(s, t)$  if  $s$  does not have a translation in  $\mathcal{L}_e$ , and recall is not penalized for failing to recover multiple translations of  $s$ .

Setting	$p_{0.1}$	$p_{0.25}$	$p_{0.33}$	$p_{0.50}$	Best- $F_1$
EDITDIST	58.6	62.6	61.1	—	47.4
ORTHO	76.0	81.3	80.1	52.3	55.0
CONTEXT	<b>91.1</b>	81.3	80.2	65.3	58.0
MCCA	87.2	<b>89.7</b>	<b>89.0</b>	<b>89.7</b>	<b>72.0</b>

Table 1: Performance of EDITDIST and our model with various features sets on EN-ES-W. See section 5.

naturally produces lexicons in which each entry is associated with a weight based on the model, we can give a full precision/recall curve (see figure 3). We summarize these curves with both the best  $F_1$  over all possible thresholds and various precisions  $p_x$  at recalls  $x$ . All reported numbers exclude evaluation on the seed lexicon entries, regardless of how those seeds are derived or whether they are correct.

In all experiments, unless noted otherwise, we used a seed of size 100 obtained from  $\mathcal{L}_e$  and considered lexicons between the top  $n = 2,000$  most frequent source and target noun word types which were not in the seed lexicon; each system proposed an already-ranked one-to-one translation lexicon amongst these  $n$  words. Where applicable, we compare against the EDITDIST baseline, which solves a maximum bipartite matching problem where edge weights are normalized edit distances. We will use MCCA (for *matching CCA*) to denote our model using the optimal feature set (see section 5.3).

## 5 Features

In this section, we explore feature representations of word types in our model. Recall that  $f_S(\cdot)$  and  $f_T(\cdot)$  map source and target word types to vectors in  $\mathbb{R}^{d_S}$  and  $\mathbb{R}^{d_T}$ , respectively (see section 2). The features used in each representation are defined identically and derived only from the appropriate monolingual corpora. For a concrete example of a word type to feature vector mapping, see figure 2.

### 5.1 Orthographic Features

For closely related languages, such as English and Spanish, translation pairs often share many *orthographic* features. One direct way to capture orthographic similarity between word pairs is edit distance. Running EDITDIST (see section 4.3) on EN-

ES-W yielded 61.1  $p_{0.33}$ , but precision quickly degrades for higher recall levels (see EDITDIST in table 1). Nevertheless, when available, orthographic clues are strong indicators of translation pairs.

We can represent orthographic features of a word type  $w$  by assigning a feature to each substring of length  $\leq 3$ . Note that MCCA can learn regular orthographic correspondences between source and target words, which is something edit distance cannot capture (see table 5). Indeed, running our MCCA model with only orthographic features on EN-ES-W, labeled ORTHO in table 1, yielded 80.1  $p_{0.33}$ , a 31% error-reduction over EDITDIST in  $p_{0.33}$ .

## 5.2 Context Features

While orthographic features are clearly effective for historically related language pairs, they are more limited for other language pairs, where we need to appeal to other clues. One non-orthographic clue that word types  $s$  and  $t$  form a translation pair is that there is a strong correlation between the source words used with  $s$  and the target words used with  $t$ . To capture this information, we define *context* features for each word type  $w$ , consisting of counts of nouns which occur within a window of size 4 around  $w$ . Consider the translation pair (time, tiempo) illustrated in figure 2. As we become more confident about other translation pairs which have active period and periodico context features, we learn that translation pairs tend to jointly generate these features, which leads us to believe that time and tiempo might be generated by a common underlying concept vector (see section 2).<sup>13</sup>

Using context features alone on EN-ES-W, our MCCA model (labeled CONTEXT in table 1) yielded a 80.2  $p_{0.33}$ . It is perhaps surprising that context features alone, without orthographic information, can yield a best- $F_1$  comparable to EDITDIST.

## 5.3 Combining Features

We can of course combine context and orthographic features. Doing so yielded 89.03  $p_{0.33}$  (labeled MCCA in table 1); this represents a 46.4% error reduction in  $p_{0.33}$  over the EDITDIST baseline. For the remainder of this work, we will use MCCA to refer

<sup>13</sup>It is important to emphasize, however, that our current model does not directly relate a word type’s role as a participant in the matching to that word’s role as a context feature.

(a) Corpus Variation

Setting	$p_{0.1}$	$p_{0.25}$	$p_{0.33}$	$p_{0.50}$	Best- $F_1$
EN-ES-G	75.0	71.2	68.3	—	49.0
EN-ES-W	87.2	89.7	89.0	89.7	72.0
EN-ES-D	91.4	94.3	92.3	89.7	63.7
EN-ES-P	97.3	94.8	93.8	92.9	77.0

(b) Seed Lexicon Variation

Corpus	$p_{0.1}$	$p_{0.25}$	$p_{0.33}$	$p_{0.50}$	Best- $F_1$
EDITDIST	58.6	62.6	61.1	—	47.4
MCCA	91.4	94.3	92.3	89.7	63.7
MCCA-AUTO	91.2	90.5	91.8	77.5	61.7

(c) Language Variation

Languages	$p_{0.1}$	$p_{0.25}$	$p_{0.33}$	$p_{0.50}$	Best- $F_1$
EN-ES	91.4	94.3	92.3	89.7	63.7
EN-FR	94.5	89.1	88.3	78.6	61.9
EN-CH	60.1	39.3	26.8	—	30.8
EN-AR	70.0	50.0	31.1	—	33.1

Table 2: (a) varying type of corpora used on system performance (section 6.1), (b) using a heuristically chosen seed compared to one taken from the evaluation lexicon (section 6.2), (c) a variety of language pairs (see section 6.3).

to our model using both orthographic and context features.

## 6 Experiments

In this section we examine how system performance varies when crucial elements are altered.

### 6.1 Corpus Variation

There are many sources from which we can derive monolingual corpora, and MCCA performance depends on the degree of similarity between corpora. We explored the following levels of relationships between corpora, roughly in order of closest to most distant:

- **Same Sentences:** EN-ES-P
- **Non-Parallel Similar Content:** EN-ES-W
- **Distinct Sentences, Same Domain:** EN-ES-D
- **Unrelated Corpora:** EN-ES-G

Our results for all conditions are presented in table 2(a). The predominant trend is that system performance degraded when the corpora diverged in

content, presumably due to context features becoming less informative. However, it is notable that even in the most extreme case of disjoint corpora from different time periods and topics (e.g. EN-ES-G), we are still able to recover lexicons of reasonable accuracy.

## 6.2 Seed Lexicon Variation

All of our experiments so far have exploited a small seed lexicon which has been derived from the evaluation lexicon (see section 4.3). In order to explore system robustness to heuristically chosen seed lexicons, we automatically extracted a seed lexicon similarly to Koehn and Knight (2002): we ran EDIT-DIST on EN-ES-D and took the top 100 most confident translation pairs. Using this automatically derived seed lexicon, we ran our system on EN-ES-D as before, evaluating on the top 2,000 noun word types not included in the automatic lexicon.<sup>14</sup> Using the automated seed lexicon, and still evaluating against our Wiktionary lexicon, MCCA-AUTO yielded 91.8  $p_{0.33}$  (see table 2(b)), indicating that our system can produce lexicons of comparable accuracy with a heuristically chosen seed. We should note that this performance represents no knowledge given to the system in the form of gold seed lexicon entries.

## 6.3 Language Variation

We also explored how system performance varies for language pairs other than English-Spanish. On English-French, for the disjoint EN-FR-D corpus (described in section 4.1), MCCA yielded 88.3  $p_{0.33}$  (see table 2(c) for more performance measures). This verified that our model can work for another closely related language-pair on which no model development was performed.

One concern is how our system performs on language pairs where orthographic features are less applicable. Results on disjoint English-Chinese and English-Arabic are given as EN-CH-D and EN-AR in table 2(c), both using only context features. In these cases, MCCA yielded much lower precisions of 26.8 and 31.0  $p_{0.33}$ , respectively. For both languages, performance degraded compared to EN-ES-

<sup>14</sup>Note that the 2,000 words evaluated here were not identical to the words tested on when the seed lexicon is derived from the evaluation lexicon.

(a) English-Spanish			
Rank	Source	Target	Correct
1.	education	educación	Y
2.	pacto	pact	Y
3.	stability	estabilidad	Y
6.	corruption	corrupción	Y
7.	tourism	turismo	Y
9.	organisation	organización	Y
10.	convenience	conveniencia	Y
11.	syria	siria	Y
12.	cooperation	cooperación	Y
14.	culture	cultura	Y
21.	protocol	protocolo	Y
23.	north	norte	Y
24.	health	salud	Y
25.	action	reacción	N

(b) English-French			
Rank	Source	Target	Correct
3.	xenophobia	xénophobie	Y
4.	corruption	corruption	Y
5.	subsidiarity	subsidiarité	Y
6.	programme	programme-cadre	N
8.	traceability	traçabilité	Y

(c) English-Chinese			
Rank	Source	Target	Correct
1.	prices	价格	Y
2.	network	网络	Y
3.	population	人口	Y
4.	reporter	孙	N
5.	oil	石油	Y

Table 3: Sample output from our (a) Spanish, (b) French, and (c) Chinese systems. We present the highest confidence system predictions, where the only editing done is to ignore predictions which consist of identical source and target words.

D and EN-FR-D, presumably due in part to the lack of orthographic features. However, MCCA still achieved surprising precision at lower recall levels. For instance, at  $p_{0.1}$ , MCCA yielded 60.1 and 70.0 on Chinese and Arabic, respectively. Figure 3 shows the highest-confidence outputs in several languages.

## 6.4 Comparison To Previous Work

There has been previous work in extracting translation pairs from non-parallel corpora (Rapp, 1995; Fung, 1995; Koehn and Knight, 2002), but generally not in as extreme a setting as the one considered here. Due to unavailability of data and specificity in experimental conditions and evaluations, it is not possible to perform exact comparisons. How-

(a) Example Non-Cognate Pairs

health	salud
traceability	rastreabilidad
youth	juventud
report	informe
advantages	ventajas

(b) Interesting Incorrect Pairs

liberal	partido
Kirkhope	Gorsel
action	reacción
Albanians	Bosnia
a.m.	horas
Netherlands	Bretaña

Table 4: System analysis on EN-ES-W: (a) non-cognate pairs proposed by our system, (b) hand-selected representative errors.

(a) Orthographic Feature

Source Feat.	Closest Target Feats.	Example Translation
#st	#es, est	(statue, estatua)
ty#	ad#, d#	(felicity, felicidad)
ogy	gía, gí	(geology, geología)

(b) Context Feature

Source Feat.	Closest Context Features
party	partido, izquierda
democrat	socialistas, demócratas
beijing	pekín, kioto

Table 5: Hand selected examples of source and target features which are close in canonical space: (a) orthographic feature correspondences, (b) context features.

ever, we attempted to run an experiment as similar as possible in setup to Koehn and Knight (2002), using English Gigaword and German Europarl. In this setting, our MCCA system yielded 61.7% accuracy on the 186 most confident predictions compared to 39% reported in Koehn and Knight (2002).

## 7 Analysis

We have presented a novel generative model for bilingual lexicon induction and presented results under a variety of data conditions (section 6.1) and languages (section 6.3) showing that our system can produce accurate lexicons even in highly adverse conditions. In this section, we broadly characterize and analyze the behavior of our system.

We manually examined the top 100 errors in the

English-Spanish lexicon produced by our system on EN-ES-W. Of the top 100 errors: 21 were correct translations not contained in the Wiktionary lexicon (e.g. *pintura* to painting), 4 were purely morphological errors (e.g. *airport* to *aeropuertos*), 30 were semantically related (e.g. *basketball* to *béisbol*), 15 were words with strong orthographic similarities (e.g. *coast* to *costas*), and 30 were difficult to categorize and fell into none of these categories. Since many of our ‘errors’ actually represent valid translation pairs not contained in our extracted dictionary, we supplemented our evaluation lexicon with one automatically derived from 100k sentences of parallel Europarl data. We ran the intersected HMM word-alignment model (Liang et al., 2008) and added  $(s, t)$  to the lexicon if  $s$  was aligned to  $t$  at least three times and more than any other word. Evaluating against the union of these lexicons yielded 98.0  $p_{0.33}$ , a significant improvement over the 92.3 using only the Wiktionary lexicon. Of the true errors, the most common arose from semantically related words which had strong context feature correlations (see table 4(b)).

We also explored the relationships our model learns between features of different languages. We projected each source and target feature into the shared canonical space, and for each projected source feature we examined the closest projected target features. In table 5(a), we present some of the orthographic feature relationships learned by our system. Many of these relationships correspond to phonological and morphological regularities such as the English suffix *ing* mapping to the Spanish suffix *gía*. In table 5(b), we present context feature correspondences. Here, the broad trend is for words which are either translations or semantically related across languages to be close in canonical space.

## 8 Conclusion

We have presented a generative model for bilingual lexicon induction based on probabilistic CCA. Our experiments show that high-precision translations can be mined without any access to parallel corpora. It remains to be seen how such lexicons can be best utilized, but they invite new approaches to the statistical translation of resource-poor languages.

## References

- Francis R. Bach and Michael I. Jordan. 2006. A probabilistic interpretation of canonical correlation analysis. Technical report, University of California, Berkeley.
- Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.
- Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of arabic text: From raw text to base phrase chunks. In *HLT-NAACL*.
- Pascale Fung. 1995. Compiling bilingual lexicon entries from a non-parallel english-chinese corpus. In *Third Annual Workshop on Very Large Corpora*.
- Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *COLING-ACL*.
- David R. Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2003. Canonical correlation analysis an overview with application to learning methods. Technical Report CSD-TR-03-02, Royal Holloway University of London.
- Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*.
- P. Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA 2004*.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.
- H. W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*.
- P. Liang, D. Klein, and M. I. Jordan. 2008. Agreement-based learning. In *NIPS*.
- Reinhard Rapp. 1995. Identifying word translation in non-parallel texts. In *ACL*.
- Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*.
- N. Smith and J. Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *ACL*.
- L. G. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201.
- D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*.

# Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora

Shiqi Zhao<sup>1</sup>, Haifeng Wang<sup>2</sup>, Ting Liu<sup>1</sup>, Sheng Li<sup>1</sup>

<sup>1</sup>Harbin Institute of Technology, Harbin, China

{zhaosq, tliu, lisheng}@ir.hit.edu.cn

<sup>2</sup>Toshiba (China) Research and Development Center, Beijing, China

wanghaifeng@rdc.toshiba.com.cn

## Abstract

Paraphrase patterns are useful in paraphrase recognition and generation. In this paper, we present a pivot approach for extracting paraphrase patterns from bilingual parallel corpora, whereby the English paraphrase patterns are extracted using the sentences in a foreign language as pivots. We propose a log-linear model to compute the paraphrase likelihood of two patterns and exploit feature functions based on maximum likelihood estimation (MLE) and lexical weighting (LW). Using the presented method, we extract over 1,000,000 pairs of paraphrase patterns from 2M bilingual sentence pairs, the precision of which exceeds 67%. The evaluation results show that: (1) The pivot approach is effective in extracting paraphrase patterns, which significantly outperforms the conventional method DIRT. Especially, the log-linear model with the proposed feature functions achieves high performance. (2) The coverage of the extracted paraphrase patterns is high, which is above 84%. (3) The extracted paraphrase patterns can be classified into 5 types, which are useful in various applications.

## 1 Introduction

Paraphrases are different expressions that convey the same meaning. Paraphrases are important in plenty of natural language processing (NLP) applications, such as question answering (QA) (Lin and Pantel, 2001; Ravichandran and Hovy, 2002), machine translation (MT) (Kauchak and Barzilay, 2006; Callison-Burch et al., 2006), multi-document

summarization (McKeown et al., 2002), and natural language generation (Iordanskaja et al., 1991).

Paraphrase patterns are sets of semantically equivalent patterns, in which a pattern generally contains two parts, i.e., the pattern words and slots. For example, in the pattern “ $X$  solves  $Y$ ”, “solves” is the pattern word, while “ $X$ ” and “ $Y$ ” are slots. One can generate a text unit (phrase or sentence) by filling the pattern slots with specific words. Paraphrase patterns are useful in both paraphrase recognition and generation. In paraphrase recognition, if two text units match a pair of paraphrase patterns and the corresponding slot-fillers are identical, they can be identified as paraphrases. In paraphrase generation, a text unit that matches a pattern  $P$  can be rewritten using the paraphrase patterns of  $P$ .

A variety of methods have been proposed on paraphrase patterns extraction (Lin and Pantel, 2001; Ravichandran and Hovy, 2002; Shinyama et al., 2002; Barzilay and Lee, 2003; Ibrahim et al., 2003; Pang et al., 2003; Szpektor et al., 2004). However, these methods have some shortcomings. Especially, the precisions of the paraphrase patterns extracted with these methods are relatively low.

In this paper, we extract paraphrase patterns from bilingual parallel corpora based on a pivot approach. We assume that if two English patterns are aligned with the same pattern in another language, they are likely to be paraphrase patterns. This assumption is an extension of the one presented in (Bannard and Callison-Burch, 2005), which was used for deriving phrasal paraphrases from bilingual corpora. Our method involves three steps: (1) corpus preprocessing, including English monolingual dependency



parsing and English-foreign language word alignment, (2) aligned patterns induction, which produces English patterns along with the aligned pivot patterns in the foreign language, (3) paraphrase patterns extraction, in which paraphrase patterns are extracted based on a log-linear model.

Our contributions are as follows. Firstly, we are the first to use a pivot approach to extract paraphrase patterns from bilingual corpora, though similar methods have been used for learning phrasal paraphrases. Our experiments show that the pivot approach significantly outperforms conventional methods. Secondly, we propose a log-linear model for computing the paraphrase likelihood. Besides, we use feature functions based on maximum likelihood estimation (MLE) and lexical weighting (LW), which are effective in extracting paraphrase patterns.

Using the proposed approach, we extract over 1,000,000 pairs of paraphrase patterns from 2M bilingual sentence pairs, the precision of which is above 67%. Experimental results show that the pivot approach evidently outperforms DIRT, a well known method that extracts paraphrase patterns from monolingual corpora (Lin and Pantel, 2001). Besides, the log-linear model is more effective than the conventional model presented in (Bannard and Callison-Burch, 2005). In addition, the coverage of the extracted paraphrase patterns is high, which is above 84%. Further analysis shows that 5 types of paraphrase patterns can be extracted with our method, which can be used in multiple NLP applications.

The rest of this paper is structured as follows. Section 2 reviews related work on paraphrase patterns extraction. Section 3 presents our method in detail. We evaluate the proposed method in Section 4, and finally conclude this paper in Section 5.

## 2 Related Work

Paraphrase patterns have been learned and used in information extraction (IE) and answer extraction of QA. For example, Lin and Pantel (2001) proposed a method (DIRT), in which they obtained paraphrase patterns from a parsed monolingual corpus based on an extended distributional hypothesis, where if two paths in dependency trees tend to occur in similar contexts it is hypothesized that the meanings of the paths are similar. The examples of obtained para-

(1)	<i>X solves Y</i> <i>Y is solved by X</i> <i>X finds a solution to Y</i> .....
(2)	<i>born in &lt;ANSWER&gt; , &lt;NAME&gt;</i> <i>&lt;NAME&gt; was born on &lt;ANSWER&gt; ,</i> <i>&lt;NAME&gt; ( &lt;ANSWER&gt; -</i> .....
(3)	<i>ORGANIZATION decides <math>\phi</math></i> <i>ORGANIZATION confirms <math>\phi</math></i> .....

Table 1: Examples of paraphrase patterns extracted with the methods of Lin and Pantel (2001), Ravichandran and Hovy (2002), and Shinyama et al. (2002).

phrase patterns are shown in Table 1 (1).

Based on the same hypothesis as above, some methods extracted paraphrase patterns from the web. For instance, Ravichandran and Hovy (2002) defined a question taxonomy for their QA system. They then used hand-crafted examples of each question type as queries to retrieve paraphrase patterns from the web. For instance, for the question type “*BIRTHDAY*”, The paraphrase patterns produced by their method can be seen in Table 1 (2).

Similar methods have also been used by Ibrahim et al. (2003) and Szpektor et al. (2004). The main disadvantage of the above methods is that the precisions of the learned paraphrase patterns are relatively low. For instance, the precisions of the paraphrase patterns reported in (Lin and Pantel, 2001), (Ibrahim et al., 2003), and (Szpektor et al., 2004) are lower than 50%. Ravichandran and Hovy (2002) did not directly evaluate the precision of the paraphrase patterns extracted using their method. However, the performance of their method is dependent on the hand-crafted queries for web mining.

Shinyama et al. (2002) presented a method that extracted paraphrase patterns from multiple news articles about the same event. Their method was based on the assumption that NEs are preserved across paraphrases. Thus the method acquired paraphrase patterns from sentence pairs that share comparable NEs. Some examples can be seen in Table 1 (3).

The disadvantage of this method is that it greatly relies on the number of NEs in sentences. The preci-

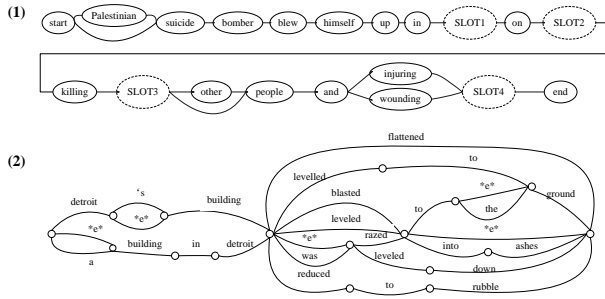


Figure 1: Examples of paraphrase patterns extracted by Barzilay and Lee (2003) and Pang et al. (2003).

sion of the extracted patterns may sharply decrease if the sentences do not contain enough NEs.

Barzilay and Lee (2003) applied multi-sequence alignment (MSA) to parallel news sentences and induced paraphrase patterns for generating new sentences (Figure 1 (1)). Pang et al. (2003) built finite state automata (FSA) from semantically equivalent translation sets based on syntactic alignment. The learned FSAs could be used in paraphrase representation and generation (Figure 1 (2)). Obviously, it is difficult for a sentence to match such complicated patterns, especially if the sentence is not from the same domain in which the patterns are extracted.

Bannard and Callison-Burch (2005) first exploited bilingual corpora for phrasal paraphrase extraction. They assumed that if two English phrases  $e_1$  and  $e_2$  are aligned with the same phrase  $c$  in another language, these two phrases may be paraphrases. Specifically, they computed the paraphrase probability in terms of the translation probabilities:

$$p(e_2|e_1) = \sum_c p_{MLE}(c|e_1)p_{MLE}(e_2|c) \quad (1)$$

In Equation (1),  $p_{MLE}(c|e_1)$  and  $p_{MLE}(e_2|c)$  are the probabilities of translating  $e_1$  to  $c$  and  $c$  to  $e_2$ , which are computed based on MLE:

$$p_{MLE}(c|e_1) = \frac{count(c, e_1)}{\sum_{c'} count(c', e_1)} \quad (2)$$

where  $count(c, e_1)$  is the frequency count that phrases  $c$  and  $e_1$  are aligned in the corpus.  $p_{MLE}(e_2|c)$  is computed in the same way.

This method proved effective in extracting high quality phrasal paraphrases. As a result, we extend it to paraphrase pattern extraction in this paper.

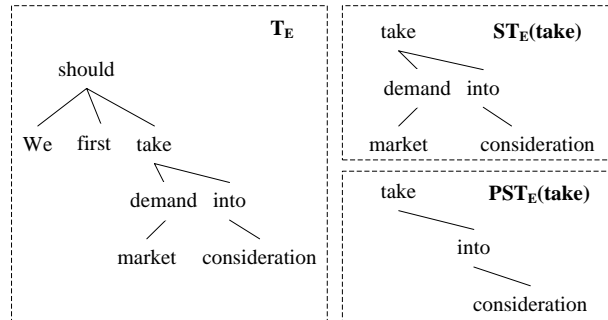


Figure 2: Examples of a subtree and a partial subtree.

### 3 Proposed Method

#### 3.1 Corpus Preprocessing

In this paper, we use English paraphrase patterns extraction as a case study. An English-Chinese (E-C) bilingual parallel corpus is employed for training. The Chinese part of the corpus is used as pivots to extract English paraphrase patterns. We conduct word alignment with Giza++ (Och and Ney, 2000) in both directions and then apply the grow-diag heuristic (Koehn et al., 2005) for symmetrization.

Since the paraphrase patterns are extracted from dependency trees, we parse the English sentences in the corpus with MaltParser (Nivre et al., 2007). Let  $S_E$  be an English sentence,  $T_E$  the parse tree of  $S_E$ ,  $e$  a word of  $S_E$ , we define the subtree and partial subtree following the definitions in (Ouangraoua et al., 2007). In detail, a subtree  $ST_E(e)$  is a particular connected subgraph of the tree  $T_E$ , which is rooted at  $e$  and includes all the descendants of  $e$ . A partial subtree  $PST_E(e)$  is a connected subgraph of the subtree  $ST_E(e)$ , which is rooted at  $e$  but does not necessarily include all the descendants of  $e$ . For instance, for the sentence “*We should first take market demand into consideration*”,  $ST_E(take)$  and  $PST_E(take)$  are shown in Figure 2<sup>1</sup>.

#### 3.2 Aligned Patterns Induction

To induce the aligned patterns, we first induce the English patterns using the subtrees and partial subtrees. Then, we extract the pivot Chinese patterns aligning to the English patterns.

<sup>1</sup>Note that, a subtree may contain several partial subtrees. In this paper, all the possible partial subtrees are considered when extracting paraphrase patterns.

**Algorithm 1: Inducing an English pattern**

```

1: Input: words in  $ST_E(e) : w_i w_{i+1} \dots w_j$ 
2: Input:  $P_E(e) = \phi$ 
3: For each  $w_k (i \leq k \leq j)$ 
4:   If  $w_k$  is in  $PST_E(e)$ 
5:     Append  $w_k$  to the end of  $P_E(e)$ 
6:   Else
7:     Append  $POS(w_k)$  to the end of  $P_E(e)$ 
8: End For

```

**Algorithm 2: Inducing an aligned pivot pattern**

```

1: Input:  $S_C = t_1 t_2 \dots t_n$ 
2: Input:  $P_C = \phi$ 
3: For each  $t_l (1 \leq l \leq n)$ 
4:   If  $t_l$  is aligned with  $w_k$  in  $S_E$ 
5:     If  $w_k$  is a word in  $P_E(e)$ 
6:       Append  $t_l$  to the end of  $P_C$ 
7:     If  $POS(w_k)$  is a slot in  $P_E(e)$ 
8:       Append  $POS(w_k)$  to the end of  $P_C$ 
9: End For

```

**Step-1 Inducing English patterns.** In this paper, an English pattern  $P_E(e)$  is a string comprising words and part-of-speech (POS) tags. Our intuition for inducing an English pattern is that a partial subtree  $PST_E(e)$  can be viewed as a unit that conveys a definite meaning, though the words in  $PST_E(e)$  may not be continuous. For example,  $PST_E(take)$  in Figure 2 contains words “take ... into consideration”. Therefore, we may extract “take X into consideration” as a pattern. In addition, the words that are in  $ST_E(e)$  but not in  $PST_E(e)$  (denoted as  $ST_E(e)/PST_E(e)$ ) are also useful for inducing patterns, since they can constrain the pattern slots. In the example in Figure 2, the word “demand” indicates that a noun can be filled in the slot X and the pattern may have the form “take NN into consideration”. Based on this intuition, we induce an English pattern  $P_E(e)$  as in Algorithm 1<sup>2</sup>.

For the example in Figure 2, the generated pattern  $P_E(take)$  is “take NN NN into consideration”. Note that the patterns induced in this way are quite specific, since the POS of each word in  $ST_E(e)/PST_E(e)$  forms a slot. Such patterns are difficult to be matched in applications. We there-

<sup>2</sup> $POS(w_k)$  in Algorithm 1 denotes the POS tag of  $w_k$ .

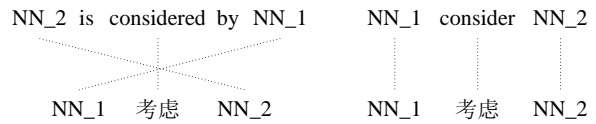


Figure 3: Aligned patterns with numbered slots.

fore take an additional step to simplify the patterns. Let  $e_i$  and  $e_j$  be two words in  $ST_E(e)/PST_E(e)$ , whose POS  $pos_i$  and  $pos_j$  are slots in  $P_E(e)$ . If  $e_i$  is a descendant of  $e_j$  in the parse tree, we remove  $pos_i$  from  $P_E(e)$ . For the example above, the POS of “market” is removed, since it is the descendant of “demand”, whose POS also forms a slot. The simplified pattern is “take NN into consideration”.

**Step-2 Extracting pivot patterns.** For each English pattern  $P_E(e)$ , we extract an aligned Chinese pivot pattern  $P_C$ . Let a Chinese sentence  $S_C$  be the translation of the English sentence  $S_E$ ,  $P_E(e)$  a pattern induced from  $S_E$ , we extract the pivot pattern  $P_C$  aligning to  $P_E(e)$  as in Algorithm 2. Note that the Chinese patterns are not extracted from parse trees. They are only sequences of Chinese words and POSes that are aligned with English patterns.

A pattern may contain two or more slots sharing the same POS. To distinguish them, we assign a number to each slot in the aligned E-C patterns. In detail, the slots having identical POS in  $P_C$  are numbered incrementally (i.e., 1,2,3...), while each slot in  $P_E(e)$  is assigned the same number as its aligned slot in  $P_C$ . The examples of the aligned patterns with numbered slots are illustrated in Figure 3.

### 3.3 Paraphrase Patterns Extraction

As mentioned above, if patterns  $e_1$  and  $e_2$  are aligned with the same pivot pattern  $c$ ,  $e_1$  and  $e_2$  may be paraphrase patterns. The paraphrase likelihood can be computed using Equation (1). However, we find that using only the MLE based probabilities can suffer from data sparseness. In order to exploit more and richer information to estimate the paraphrase likelihood, we propose a log-linear model:

$$score(e_2|e_1) = \sum_c \exp\left[\sum_{i=1}^N \lambda_i h_i(e_1, e_2, c)\right] \quad (3)$$

where  $h_i(e_1, e_2, c)$  is a feature function and  $\lambda_i$  is the

weight. In this paper, 4 feature functions are used in our log-linear model, which include:

$$\begin{aligned} h_1(e_1, e_2, c) &= score_{MLE}(c|e_1) \\ h_2(e_1, e_2, c) &= score_{MLE}(e_2|c) \\ h_3(e_1, e_2, c) &= score_{LW}(c|e_1) \\ h_4(e_1, e_2, c) &= score_{LW}(e_2|c) \end{aligned}$$

Feature functions  $h_1(e_1, e_2, c)$  and  $h_2(e_1, e_2, c)$  are based on MLE.  $score_{MLE}(c|e)$  is computed as:

$$score_{MLE}(c|e) = \log p_{MLE}(c|e) \quad (4)$$

$score_{MLE}(e|c)$  is computed in the same way.

$h_3(e_1, e_2, c)$  and  $h_4(e_1, e_2, c)$  are based on LW. LW was originally used to validate the quality of a phrase translation pair in MT (Koehn et al., 2003). It checks how well the words of the phrases translate to each other. This paper uses LW to measure the quality of aligned patterns. We define  $score_{LW}(c|e)$  as the logarithm of the lexical weight<sup>3</sup>:

$$score_{LW}(c|e) = \frac{1}{n} \sum_{i=1}^n \log \left( \frac{1}{|\{j|(i, j) \in a\}|} \sum_{\forall (i, j) \in a} w(c_i|e_j) \right) \quad (5)$$

where  $a$  denotes the word alignment between  $c$  and  $e$ .  $n$  is the number of words in  $c$ .  $c_i$  and  $e_j$  are words of  $c$  and  $e$ .  $w(c_i|e_j)$  is computed as follows:

$$w(c_i|e_j) = \frac{count(c_i, e_j)}{\sum_{c'_i} count(c'_i, e_j)} \quad (6)$$

where  $count(c_i, e_j)$  is the frequency count of the aligned word pair  $(c_i, e_j)$  in the corpus.  $score_{LW}(e|c)$  is computed in the same manner.

In our experiments, we set a threshold  $T$ . If the score between  $e_1$  and  $e_2$  based on Equation (3) exceeds  $T$ ,  $e_2$  is extracted as the paraphrase of  $e_1$ .

### 3.4 Parameter Estimation

Five parameters need to be estimated, i.e.,  $\lambda_1$ ,  $\lambda_2$ ,  $\lambda_3$ ,  $\lambda_4$  in Equation (3), and the threshold  $T$ . To estimate the parameters, we first construct a development set. In detail, we randomly sample 7,086

<sup>3</sup>The logarithm of the lexical weight is divided by  $n$  so as not to penalize long patterns.

groups of aligned E-C patterns that are obtained as described in Section 3.2. The English patterns in each group are all aligned with the same Chinese pivot pattern. We then extract paraphrase patterns from the aligned patterns as described in Section 3.3. In this process, we set  $\lambda_i = 1$  ( $i = 1, \dots, 4$ ) and assign  $T$  a minimum value, so as to obtain all possible paraphrase patterns.

A total of 4,162 pairs of paraphrase patterns have been extracted and manually labeled as “1” (correct paraphrase patterns) or “0” (incorrect). Here, two patterns are regarded as paraphrase patterns if they can generate paraphrase fragments by filling the corresponding slots with identical words. We use gradient descent algorithm (Press et al., 1992) to estimate the parameters. For each set of parameters, we compute the precision  $P$ , recall  $R$ , and f-measure  $F$  as:  $P = \frac{|set1 \cap set2|}{|set1|}$ ,  $R = \frac{|set1 \cap set2|}{|set2|}$ ,  $F = \frac{2PR}{P+R}$ , where  $set1$  denotes the set of paraphrase patterns extracted under the current parameters.  $set2$  denotes the set of manually labeled correct paraphrase patterns. We select the parameters that can maximize the F-measure on the development set<sup>4</sup>.

## 4 Experiments

The E-C parallel corpus in our experiments was constructed using several LDC bilingual corpora<sup>5</sup>. After filtering sentences that are too long ( $> 40$  words) or too short ( $< 5$  words), 2,048,009 pairs of parallel sentences were retained.

We used two constraints in the experiments to improve the efficiency of computation. First, only subtrees containing no more than 10 words were used to induce English patterns. Second, although any POS tag can form a slot in the induced patterns, we only focused on three kinds of POSes in the experiments, i.e., nouns (tags include NN, NNS, NNP, NNPS), verbs (VB, VBD, VBG, VBN, VBP, VBZ), and adjectives (JJ, JJS, JJR). In addition, we constrained that a pattern must contain at least one content word

<sup>4</sup>The parameters are:  $\lambda_1 = 0.0594137$ ,  $\lambda_2 = 0.995936$ ,  $\lambda_3 = -0.0048954$ ,  $\lambda_4 = 1.47816$ ,  $T = -10.002$ .

<sup>5</sup>The corpora include LDC2000T46, LDC2000T47, LDC2002E18, LDC2002T01, LDC2003E07, LDC2003E14, LDC2003T17, LDC2004E12, LDC2004T07, LDC2004T08, LDC2005E83, LDC2005T06, LDC2005T10, LDC2006E24, LDC2006E34, LDC2006E85, LDC2006E92, LDC2006T04, LDC2007T02, LDC2007T09.

Method	#PP (pairs)	Precision
LL-Model	1,058,624	67.03%
MLE-Model	1,015,533	60.60%
DIRT top-1	1,179	19.67%
DIRT top-5	5,528	18.73%

Table 2: Comparison of paraphrasing methods.

so as to filter patterns like “*the [NN\_I]*”.

#### 4.1 Evaluation of the Log-linear Model

As previously mentioned, in the log-linear model of this paper, we use both MLE based and LW based feature functions. In this section, we evaluate the log-linear model (LL-Model) and compare it with the MLE based model (MLE-Model) presented by Bannard and Callison-Burch (2005)<sup>6</sup>.

We extracted paraphrase patterns using two models, respectively. From the results of each model, we randomly picked 3,000 pairs of paraphrase patterns to evaluate the precision. The 6,000 pairs of paraphrase patterns were mixed and presented to the human judges, so that the judges cannot know by which model each pair was produced. The sampled patterns were then manually labeled and the precision was computed as described in Section 3.4.

The number of the extracted paraphrase patterns (#PP) and the precision are depicted in the first two lines of Table 2. We can see that the numbers of paraphrase patterns extracted using the two models are comparable. However, the precision of LL-Model is significantly higher than MLE-Model.

Actually, MLE-Model is a special case of LL-Model and the enhancement of the precision is mainly due to the use of LW based features. It is not surprising, since Bannard and Callison-Burch (2005) have pointed out that word alignment error is the major factor that influences the performance of the methods learning paraphrases from bilingual corpora. The LW based features validate the quality of word alignment and assign low scores to those aligned E-C pattern pairs with incorrect alignment. Hence the precision can be enhanced.

<sup>6</sup>In this experiment, we also estimated a threshold  $T'$  for MLE-Model using the development set ( $T' = -5.1$ ). The pattern pairs whose score based on Equation (1) exceed  $T'$  were extracted as paraphrase patterns.

#### 4.2 Comparison with DIRT

It is necessary to compare our method with another paraphrase patterns extraction method. However, it is difficult to find methods that are suitable for comparison. Some methods only extract paraphrase patterns using news articles on certain topics (Shinyama et al., 2002; Barzilay and Lee, 2003), while some others need seeds as initial input (Ravichandran and Hovy, 2002). In this paper, we compare our method with DIRT (Lin and Pantel, 2001), which does not need to specify topics or input seeds.

As mentioned in Section 2, DIRT learns paraphrase patterns from a parsed monolingual corpus based on an extended distributional hypothesis. In our experiment, we implemented DIRT and extracted paraphrase patterns from the English part of our bilingual parallel corpus. Our corpus is smaller than that reported in (Lin and Pantel, 2001). To alleviate the data sparseness problem, we only kept patterns appearing more than 10 times in the corpus for extracting paraphrase patterns. Different from our method, no threshold was set in DIRT. Instead, the extracted paraphrase patterns were ranked according to their scores. In our experiment, we kept top-5 paraphrase patterns for each target pattern.

From the extracted paraphrase patterns, we sampled 600 groups for evaluation. Each group comprises a target pattern and its top-5 paraphrase patterns. The sampled data were manually labeled and the top-n precision was calculated as  $\frac{\sum_{i=1}^N n_i}{N \times n}$ , where  $N$  is the number of groups and  $n_i$  is the number of correct paraphrase patterns in the top-n paraphrase patterns of the  $i$ -th group. The top-1 and top-5 results are shown in the last two lines of Table 2. Although there are more correct patterns in the top-5 results, the precision drops sequentially from top-1 to top-5 since the denominator of top-5 is 4 times larger than that of top-1.

Obviously, the number of the extracted paraphrase patterns is much smaller than that extracted using our method. Besides, the precision is also much lower. We believe that there are two reasons. First, the extended distributional hypothesis is not strict enough. Patterns sharing similar slot-fillers do not necessarily have the same meaning. They may even have the opposite meanings. For example, “*X worsens Y*” and “*X solves Y*” were extracted as para-

Type	Count	Example
trivial change	79	(e <sub>1</sub> ) <i>all the members of [NNPS_1]</i> (e <sub>2</sub> ) <i>all members of [NNPS_1]</i>
phrase replacement	267	(e <sub>1</sub> ) <i>[JJ_1] economic losses</i> (e <sub>2</sub> ) <i>[JJ_1] financial losses</i>
phrase reordering	56	(e <sub>1</sub> ) <i>[NN_1] definition</i> (e <sub>2</sub> ) <i>the definition of [NN_1]</i>
structural paraphrase	71	(e <sub>1</sub> ) <i>the admission of [NNP_1] to the wto</i> (e <sub>2</sub> ) <i>the [NNP_1] 's wto accession</i>
information + or -	27	(e <sub>1</sub> ) <i>[NNS_1] are in fact women</i> (e <sub>2</sub> ) <i>[NNS_1] are women</i>

Table 3: The statistics and examples of each type of paraphrase patterns.

phrase patterns by DIRT. The other reason is that DIRT can only be effective for patterns appearing plenty of times in the corpus. In other words, it seriously suffers from data sparseness. We believe that DIRT can perform better on a larger corpus.

### 4.3 Pivot Pattern Constraints

As described in Section 3.2, we constrain that the pattern words of an English pattern  $e$  must be extracted from a partial subtree. However, we do not have such constraint on the Chinese pivot patterns. Hence, it is interesting to investigate whether the performance can be improved if we constrain that the pattern words of a pivot pattern  $c$  must also be extracted from a partial subtree.

To conduct the evaluation, we parsed the Chinese sentences of the corpus with a Chinese dependency parser (Liu et al., 2006). We then induced English patterns and extracted aligned pivot patterns. For the aligned patterns ( $e, c$ ), if  $c$ 's pattern words were not extracted from a partial subtree, the pair was filtered. After that, we extracted paraphrase patterns, from which we sampled 3,000 pairs for evaluation.

The results show that 736,161 pairs of paraphrase patterns were extracted and the precision is 65.77%. Compared with Table 2, the number of the extracted paraphrase patterns gets smaller and the precision also gets lower. The results suggest that the performance of the method cannot be improved by constraining the extraction of pivot patterns.

### 4.4 Analysis of the Paraphrase Patterns

We sampled 500 pairs of correct paraphrase patterns extracted using our method and analyzed the types. We found that there are 5 types of paraphrase patterns, which include: (1) trivial change, such as changes of prepositions and articles, etc; (2) phrase replacement; (3) phrase reordering; (4) struc-

tural paraphrase, which contain both phrase replacements and phrase reordering; (5) adding or reducing information that does not change the meaning. Some statistics and examples are shown in Table 3.

The paraphrase patterns are useful in NLP applications. Firstly, over 50% of the paraphrase patterns are in the type of phrase replacement, which can be used in IE pattern reformulation and sentence-level paraphrase generation. Compared with phrasal paraphrases, the phrase replacements in patterns are more accurate due to the constraints of the slots.

The paraphrase patterns in the type of phrase reordering can also be used in IE pattern reformulation and sentence paraphrase generation. Especially, in sentence paraphrase generation, this type of paraphrase patterns can reorder the phrases in a sentence, which can hardly be achieved by the conventional MT-based generation method (Quirk et al., 2004).

The structural paraphrase patterns have the advantages of both phrase replacement and phrase reordering. More paraphrase sentences can be generated using these patterns.

The paraphrase patterns in the type of “information + and -” are useful in sentence compression and expansion. A sentence matching a long pattern can be compressed by paraphrasing it using shorter patterns. Similarly, a short sentence can be expanded by paraphrasing it using longer patterns.

For the 3,000 pairs of test paraphrase patterns, we also investigate the number and type of the pattern slots. The results are summarized in Table 4 and 5.

From Table 4, we can see that more than 92% of the paraphrase patterns contain only one slot, just like the examples shown in Table 3. In addition, about 7% of the paraphrase patterns contain two slots, such as “*give [NN\_1] [NN\_2]*” vs. “*give [NN\_2] to [NN\_1]*”. This result suggests that our method tends to extract short paraphrase patterns,

Slot No.	#PP	Percentage	Precision
1-slot	2,780	92.67%	66.51%
2-slots	218	7.27%	73.85%
≥3-slots	2	<1%	50.00%

Table 4: The statistics of the numbers of pattern slots.

Slot Type	#PP	Percentage	Precision
N-slots	2,376	79.20%	66.71%
V-slots	273	9.10%	70.33%
J-slots	438	14.60%	70.32%

Table 5: The statistics of the type of pattern slots.

which is mainly because the data sparseness problem is more serious when extracting long patterns.

From Table 5, we can find that near 80% of the paraphrase patterns contain noun slots, while about 9% and 15% contain verb slots and adjective slots<sup>7</sup>. This result implies that nouns are the most typical variables in paraphrase patterns.

#### 4.5 Evaluation within Context Sentences

In Section 4.1, we have evaluated the precision of the paraphrase patterns without considering context information. In this section, we evaluate the paraphrase patterns within specific context sentences.

The open test set includes 119 English sentences. We parsed the sentences with MaltParser and induced patterns as described in Section 3.2. For each pattern  $e$  in sentence  $S_E$ , we searched  $e$ 's paraphrase patterns from the database of the extracted paraphrase patterns. The result shows that 101 of the 119 sentences contain at least one pattern that can be paraphrased using the extracted paraphrase patterns, the coverage of which is 84.87%.

Furthermore, since a pattern may have several paraphrase patterns, we exploited a method to automatically select the best one in the given context sentence. In detail, a paraphrase pattern  $e'$  of  $e$  was reranked based on a language model (LM):

$$\begin{aligned} score(e'|e, S_E) = \\ \lambda score_{LL}(e'|e) + (1 - \lambda) score_{LM}(e'|S_E) \end{aligned} \quad (7)$$

<sup>7</sup>Notice that, a pattern may contain more than one type of slots, thus the sum of the percentages is larger than 1.

Here,  $score_{LL}(e'|e)$  denotes the score based on Equation (3).  $score_{LM}(e'|S_E)$  is the LM based score:  $score_{LM}(e'|S_E) = \frac{1}{n} \log P_{LM}(S'_E)$ , where  $S'_E$  is the sentence generated by replacing  $e$  in  $S_E$  with  $e'$ . The language model in the experiment was a tri-gram model trained using the English sentences in the bilingual corpus. We empirically set  $\lambda = 0.7$ .

The selected best paraphrase patterns in context sentences were manually labeled. The context information was also considered by our judges. The result shows that the precision of the best paraphrase patterns is 59.39%. To investigate the contribution of the LM based score, we ran the experiment again with  $\lambda = 1$  (ignoring the LM based score) and found that the precision is 57.09%. It indicates that the LM based reranking can improve the precision. However, the improvement is small. Further analysis shows that about 70% of the correct paraphrase substitutes are in the type of phrase replacement.

## 5 Conclusion

This paper proposes a pivot approach for extracting paraphrase patterns from bilingual corpora. We use a log-linear model to compute the paraphrase likelihood and exploit feature functions based on MLE and LW. Experimental results show that the pivot approach is effective, which extracts over 1,000,000 pairs of paraphrase patterns from 2M bilingual sentence pairs. The precision and coverage of the extracted paraphrase patterns exceed 67% and 84%, respectively. In addition, the log-linear model with the proposed feature functions significantly outperforms the conventional models. Analysis shows that 5 types of paraphrase patterns are extracted with our method, which are useful in various applications.

In the future we wish to exploit more feature functions in the log-linear model. In addition, we will try to make better use of the context information when replacing paraphrase patterns in context sentences.

## Acknowledgments

This research was supported by National Natural Science Foundation of China (60503072, 60575042). We thank Lin Zhao, Xiaohang Qu, and Zhenghua Li for their help in the experiments.

## References

- Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, pages 597-604.
- Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of HLT-NAACL*, pages 16-23.
- Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Translation Using Paraphrases. In *Proceedings of HLT-NAACL*, pages 17-24.
- Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting Structural Paraphrases from Aligned Monolingual Corpora. In *Proceedings of IWP*, pages 57-64.
- Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. 1991. Lexical Selection and Paraphrase in a Meaning-Text Generation Model. In Cécile L. Paris, William R. Swartout, and William C. Mann (Eds.): *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 293-312.
- David Kauchak and Regina Barzilay. 2006. Paraphrasing for Automatic Evaluation. In *Proceedings of HLT-NAACL*, pages 455-462.
- Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *Proceedings of IWSLT*.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL*, pages 127-133.
- De-Kang Lin and Patrick Pantel. 2001. Discovery of Inference Rules for Question Answering. In *Natural Language Engineering* 7(4): 343-360.
- Ting Liu, Jin-Shan Ma, Hui-Jia Zhu, and Sheng Li. 2006. Dependency Parsing Based on Dynamic Local Optimization. In *Proceedings of CoNLL-X*, pages 211-215.
- Kathleen R. Mckeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L. Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. 2002. Tracking and Summarizing News on a Daily Basis with Columbia's Newsblaster. In *Proceedings of HLT*, pages 280-285.
- Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. In *Natural Language Engineering* 13(2): 95-135.
- Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of ACL*, pages 440-447.
- Aïda Ouangraoua, Pascal Ferraro, Laurent Tichit, and Serge Dulucq. 2007. Local Similarity between Quotiented Ordered Trees. In *Journal of Discrete Algorithms* 5(1): 23-35.
- Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based Alignment of Multiple Translations: Extracting Paraphrases and Generating New Sentences. In *Proceedings of HLT-NAACL*, pages 102-109.
- William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, U.K., 1992, 412-420.
- Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of EMNLP*, pages 142-149.
- Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of ACL*, pages 41-47.
- Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic Paraphrase Acquisition from News Articles. In *Proceedings of HLT*, pages 40-46.
- Idan Szepktor, Hristo Tanev, Ido Dagan and Bonaventura Coppola. 2004. Scaling Web-based Acquisition of Entailment Relations. In *Proceedings of EMNLP*, pages 41-48.



# Unsupervised Learning of Narrative Event Chains

Nathanael Chambers and Dan Jurafsky

Department of Computer Science

Stanford University

Stanford, CA 94305

{natec, jurafsky}@stanford.edu

## Abstract

Hand-coded *scripts* were used in the 1970-80s as knowledge backbones that enabled inference and other NLP tasks requiring deep semantic knowledge. We propose unsupervised induction of similar schemata called *narrative event chains* from raw newswire text.

A narrative event chain is a partially ordered set of events related by a common protagonist. We describe a three step process to learning narrative event chains. The first uses unsupervised distributional methods to learn narrative relations between events sharing corefering arguments. The second applies a temporal classifier to partially order the connected events. Finally, the third prunes and clusters self-contained chains from the space of events. We introduce two evaluations: the *narrative cloze* to evaluate event relatedness, and an *order coherence* task to evaluate narrative order. We show a 36% improvement over baseline for narrative prediction and 25% for temporal coherence.

## 1 Introduction

This paper induces a new representation of structured knowledge called **narrative event chains** (or narrative chains). Narrative chains are partially ordered sets of events centered around a common **protagonist**. They are related to structured sequences of participants and events that have been called **scripts** (Schank and Abelson, 1977) or *Fillmorean frames*. These participants and events can be filled in and instantiated in a particular text situation to draw inferences. Chains focus on a single actor to facili-

tate learning, and thus this paper addresses the three tasks of chain induction: *narrative event induction*, *temporal ordering of events* and *structured selection* (pruning the event space into discrete sets).

Learning these prototypical schematic sequences of events is important for rich understanding of text. Scripts were central to natural language understanding research in the 1970s and 1980s for proposed tasks such as summarization, coreference resolution and question answering. For example, Schank and Abelson (1977) proposed that understanding text about restaurants required knowledge about the Restaurant Script, including the participants (Customer, Waiter, Cook, Tables, etc.), the events constituting the script (entering, sitting down, asking for menus, etc.), and the various preconditions, ordering, and results of each of the constituent actions.

Consider these two distinct narrative chains.

- <b>accused X</b>		<b>W joined</b> -
<b>X claimed</b> -		<b>W served</b> -
<b>X argued</b>		<b>W oversaw</b> -
- <b>dismissed X</b>		<b>W resigned</b>

It would be useful for question answering or textual entailment to know that ‘**X denied** -’ is also a likely event in the left chain, while ‘- **replaces W**’ temporally follows the right. Narrative chains (such as *Firing of Employee* or *Executive Resigns*) offer the structure and power to directly infer these new subevents by providing critical background knowledge. In part due to its complexity, automatic induction has not been addressed since the early non-statistical work of Mooney and DeJong (1985).

The first step to narrative induction uses an entity-based model for learning narrative relations by fol-

lowing a protagonist. As a narrative progresses through a series of events, each event is characterized by the grammatical role played by the protagonist, and by the protagonist’s shared connection to surrounding events. Our algorithm is an unsupervised distributional learning approach that uses coreferring arguments as evidence of a narrative relation. We show, using a new evaluation task called **narrative cloze**, that our protagonist-based method leads to better induction than a verb-only approach.

The next step is to order events in the same narrative chain. We apply work in the area of temporal classification to create partial orders of our learned events. We show, using a coherence-based evaluation of temporal ordering, that our partial orders lead to better coherence judgements of real narrative instances extracted from documents.

Finally, the space of narrative events and temporal orders is clustered and pruned to create discrete sets of narrative chains.

## 2 Previous Work

While previous work hasn’t focused specifically on learning narratives<sup>1</sup>, our work draws from two lines of research in summarization and anaphora resolution. In summarization, **topic signatures** are a set of terms indicative of a topic (Lin and Hovy, 2000). They are extracted from hand-sorted (by topic) sets of documents using log-likelihood ratios. These terms can capture some narrative relations, but the model requires topic-sorted training data.

Bean and Riloff (2004) proposed the use of **caseframe networks** as a kind of contextual role knowledge for anaphora resolution. A caseframe is a verb/event and a semantic role (e.g. *<patient> kidnapped*). Caseframe networks are relations between caseframes that may represent synonymy (*<patient> kidnapped* and *<patient> abducted*) or related events (*<patient> kidnapped* and *<patient> released*). Bean and Riloff learn these networks from two topic-specific texts and apply them to the problem of anaphora resolution. Our work can be seen as an attempt to generalize the intuition of caseframes (finding an entire set of events

rather than just pairs of related frames) and apply it to a different task (finding a coherent structured narrative in non-topic-specific text).

More recently, Brody (2007) proposed an approach similar to caseframes that discovers high-level relatedness between verbs by grouping verbs that share the same lexical items in subject/object positions. He calls these shared arguments *anchors*. Brody learns pairwise relations between clusters of related verbs, similar to the results with caseframes. A human evaluation of these pairs shows an improvement over baseline. This and previous caseframe work lend credence to learning relations from verbs with common arguments.

We also draw from lexical chains (Morris and Hirst, 1991), indicators of text coherence from word overlap/similarity. We use a related notion of protagonist overlap to motivate narrative chain learning.

Work on semantic similarity learning such as Chklovski and Pantel (2004) also automatically learns relations between verbs. We use similar distributional scoring metrics, but differ with our use of a protagonist as the indicator of relatedness. We also use typed dependencies and the entire space of events for similarity judgements, rather than only pairwise lexical decisions.

Finally, Fujiki et al. (2003) investigated script acquisition by extracting the 41 most frequent pairs of events from the first paragraph of newswire articles, using the assumption that the paragraph’s textual order follows temporal order. Our model, by contrast, learns entire event chains, uses more sophisticated probabilistic measures, and uses temporal ordering models instead of relying on document order.

## 3 The Narrative Chain Model

### 3.1 Definition

Our model is inspired by Centering (Grosz et al., 1995) and other entity-based models of coherence (Barzilay and Lapata, 2005) in which an entity is in focus through a sequence of sentences. We propose to use this same intuition to induce narrative chains.

We assume that although a narrative has several participants, there is a central actor who characterizes a narrative chain: the **protagonist**. Narrative chains are thus structured by the protagonist’s grammatical roles in the events. In addition, narrative

<sup>1</sup>We analyzed FrameNet (Baker et al., 1998) for insight, but found that very few of the frames are event *sequences* of the type characterizing narratives and scripts.

events are ordered by some theory of time. This paper describes a partial ordering with the *before* (no overlap) relation.

Our task, therefore, is to learn events that constitute narrative chains. Formally, a **narrative chain** is a partially ordered set of narrative events that share a common actor. A **narrative event** is a tuple of an event (most simply a verb) and its participants, represented as *typed dependencies*. Since we are focusing on a single actor in this study, a narrative event is thus a tuple of the event and the typed dependency of the protagonist: (*event, dependency*). A narrative chain is a set of narrative events  $\{e_1, e_2, \dots, e_n\}$ , where  $n$  is the size of the chain, and a relation  $B(e_i, e_j)$  that is true if narrative event  $e_i$  occurs strictly before  $e_j$  in time.

### 3.2 The Protagonist

The notion of a protagonist motivates our approach to narrative learning. We make the following assumption of **narrative coherence**: *verbs sharing coreferring arguments are semantically connected by virtue of narrative discourse structure*. A single document may contain more than one narrative (or topic), but the narrative assumption states that a series of argument-sharing verbs is more likely to participate in a narrative chain than those not sharing.

In addition, the narrative approach captures grammatical constraints on *narrative coherence*. Simple distributional learning might discover that the verb *push* is related to the verb *fall*, but narrative learning can capture additional facts about the participants, specifically, that the object or patient of the *push* is the subject or agent of the *fall*.

Each focused protagonist chain offers one perspective on a narrative, similar to the multiple perspectives on a commercial transaction event offered by buy and sell.

### 3.3 Partial Ordering

A narrative chain, by definition, includes a partial ordering of events. Early work on scripts included ordering constraints with more complex preconditions and side effects on the sequence of events. This paper presents work toward a partial ordering and leaves logical constraints as future work. We focus on the *before* relation, but the model does not preclude advanced theories of temporal order.

## 4 Learning Narrative Relations

Our first model learns basic information about a narrative chain: the protagonist and the constituent subevents, although not their ordering. For this we need a metric for the relation between an event and a narrative chain.

Pairwise relations between events are first extracted unsupervised. A distributional score based on how often two events share grammatical arguments (using pointwise mutual information) is used to create this pairwise relation. Finally, a global narrative score is built such that *all* events in the chain provide feedback on the event in question (whether for inclusion or for decisions of inference).

Given a list of observed verb/dependency counts, we approximate the pointwise mutual information (PMI) by:

$$pmi(e(w, d), e(v, g)) = \log \frac{P(e(w, d), e(v, g))}{P(e(w, d))P(e(v, g))} \quad (1)$$

where  $e(w, d)$  is the verb/dependency pair  $w$  and  $d$  (e.g.  $e(\text{push}, \text{subject})$ ). The numerator is defined by:

$$P(e(w, d), e(v, g)) = \frac{C(e(w, d), e(v, g))}{\sum_{x,y} \sum_{d,f} C(e(x, d), e(y, f))} \quad (2)$$

where  $C(e(x, d), e(y, f))$  is the number of times the two events  $e(x, d)$  and  $e(y, f)$  had a coreferring entity filling the values of the dependencies  $d$  and  $f$ . We also adopt the ‘discount score’ to penalize low occurring words (Pantel and Ravichandran, 2004).

Given the debate over appropriate metrics for distributional learning, we also experimented with the t-test. Our experiments found that PMI outperforms the t-test on this task by itself and when interpolated together using various mixture weights.

Once pairwise relation scores are calculated, a global narrative score can then be built such that *all* events provide feedback on the event in question. For instance, given all narrative events in a document, we can find the next most likely event to occur by maximizing:

$$\max_{j:0 < j < m} \sum_{i=0}^n pmi(e_i, f_j) \quad (3)$$

where  $n$  is the number of events in our chain and  $e_i$  is the  $i$ th event.  $m$  is the number of events  $f$  in our training corpus. A ranked list of guesses can be built from this summation and we hypothesize that

<b>Known events:</b>			
(pleaded subj), (admits subj), (convicted obj)			
<b>Likely Events:</b>			
sentenced obj	0.89	indicted obj	0.74
paroled obj	0.76	finned obj	0.73
fired obj	0.75	denied subj	0.73

Figure 1: Three narrative events and the six most likely events to include in the same chain.

the more events in our chain, the more informed our ranked output. An example of a chain with 3 events and the top 6 ranked guesses is given in figure 1.

#### 4.1 Evaluation Metric: Narrative Cloze

The **cloze task** (Taylor, 1953) is used to evaluate a system (or human) for language proficiency by removing a random word from a sentence and having the system attempt to fill in the blank (e.g. *I forgot to \_\_\_ the waitress for the good service*). Depending on the type of word removed, the test can evaluate syntactic knowledge as well as semantic. Deyes (1984) proposed an extended task, **discourse cloze**, to evaluate discourse knowledge (removing phrases that are recoverable from knowledge of discourse relations like *contrast* and *consequence*).

We present a new cloze task that requires narrative knowledge to solve, the **narrative cloze**. The narrative cloze is a sequence of narrative events in a document from which one event has been removed. The task is to predict the missing verb and typed dependency. Take this example text about American football with McCann as the protagonist:

1. McCann threw two interceptions early.
2. Toledo pulled McCann aside and told him he'd start.
3. McCann quickly completed his first two passes.

These clauses are represented in the narrative model as five events: **(threw subject)**, **(pulled object)**, **(told object)**, **(start subject)**, **(completed subject)**. These verb/dependency events make up a narrative cloze model. We could remove **(threw subject)** and use the remaining four events to rank this missing event. Removing a single such pair to be filled in automatically allows us to evaluate a system's knowledge of narrative relations and coherence. We do not claim this cloze task to be solvable even by humans,

#### New York Times Editorial

occupied subj    brought subj    rejecting subj  
 projects subj    met subj        appeared subj  
 offered subj     voted pp\_for    offer subj  
 thinks subj

Figure 2: One of the 69 test documents, containing 10 narrative events. The protagonist is President Bush.

but rather assert it as a comparative measure to evaluate narrative knowledge.

#### 4.2 Narrative Cloze Experiment

We use years 1994-2004 (1,007,227 documents) of the Gigaword Corpus (Graff, 2002) for **training**<sup>2</sup>. We parse the text into typed dependency graphs with the Stanford Parser (de Marneffe et al., 2006)<sup>3</sup>, recording all verbs with subject, object, or prepositional typed dependencies. We use the OpenNLP<sup>4</sup> coreference engine to resolve the entity mentions. For each document, the verb pairs that share corefering entities are recorded with their dependency types. Particles are included with the verb.

We used 10 news stories from the 1994 section of the corpus for *development*. The stories were hand chosen to represent a range of topics such as business, sports, politics, and obituaries. We used 69 news stories from the 2001 (year selected randomly) section of the corpus for **testing** (also removed from training). The test set documents were randomly chosen and not preselected for a range of topics. From each document, the entity involved in the most events was selected as the protagonist. For this evaluation, we only look at verbs. All verb clauses involving the protagonist are manually extracted and translated into the narrative events (*verb,dependency*). Exceptions that are not included are verbs in headlines, quotations (typically not part of a narrative), "be" properties (e.g. *john is happy*), modifying verbs (e.g. *hurried to leave*, only *leave* is used), and multiple instances of one event.

The original test set included 100 documents, but

<sup>2</sup>The document count does not include duplicate news stories. We found up to 18% of the corpus are duplications, mostly AP reprints. We automatically found these by matching the first two paragraphs of each document, removing exact matches.

<sup>3</sup><http://nlp.stanford.edu/software/lex-parser.shtml>

<sup>4</sup><http://opennlp.sourceforge.net>

those without a narrative chain at least five events in length were removed, leaving 69 documents. Most of the removed documents were not stories, but genres such as interviews and cooking recipes. An example of an extracted chain is shown in figure 2.

We evaluate with Narrative Cloze using leave-one-out cross validation, removing one event and using the rest to generate a ranked list of guesses. The test dataset produces 740 cloze tests (69 narratives with 740 events). After generating our ranked guesses, the position of the correct event is averaged over all 740 tests for the final score. We penalize unseen events by setting their ranked position to the length of the guess list (ranging from 2k to 15k).

Figure 1 is an example of a ranked guess list for a short chain of three events. If the original document contained (*fired obj*), this cloze test would score 3.

#### 4.2.1 Baseline

We want to measure the utility of the protagonist and the *narrative coherence assumption*, so our baseline learns relatedness strictly based upon verb co-occurrence. The PMI is then defined as between all occurrences of two verbs in the same document. This baseline evaluation is verb only, as dependencies require a protagonist to fill them.

After initial evaluations, the baseline was performing very poorly due to the huge amount of data involved in counting all possible verb pairs (using a protagonist vastly reduces the number). We experimented with various count cutoffs to remove rare occurring pairs of verbs. The final results use a baseline where all pairs occurring less than 10 times in the training data are removed.

Since the verb-only baseline does not use typed dependencies, our narrative model cannot directly compare to this abstracted approach. We thus modified the narrative model to ignore typed dependencies, but still count events with shared arguments. Thus, we calculate the PMI across verbs *that share arguments*. This approach is called **Protagonist**. The full narrative model that includes the grammatical dependencies is called **Typed Deps**.

#### 4.2.2 Results

Experiments with varying sizes of training data are presented in figure 3. Each ranked list of candidate verbs for the missing event in Base-

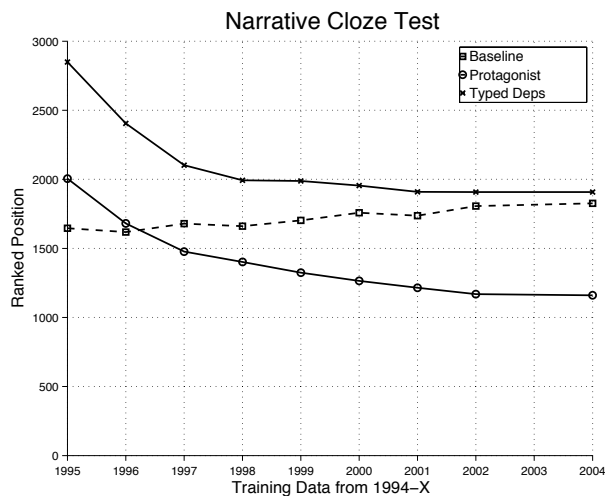


Figure 3: Results with varying sizes of training data. Year 2003 is not explicitly shown because it has an unusually small number of documents compared to other years.

line/Protagonist contained approximately 9 thousand candidates. Of the 740 cloze tests, 714 of the removed events were present in their respective list of guesses. This is encouraging as only 3.5% of the events are unseen (or do not meet cutoff thresholds).

When all training data is used (1994-2004), the average ranked position is 1826 for **Baseline** and 1160 for **Protagonist** (1 being most confident). The Baseline performs better at first (years 1994-5), but as more data is seen, the Baseline worsens while the Protagonist improves. This verb-only narrative model shows a **36.5%** improvement over the baseline trained on all years. Results from the full **Typed Deps** model, not comparable to the baseline, parallel the Protagonist results, improving as more data is seen (average ranked position of 1908 with all the training data). We also ran the experiment without OpenNLP coreference, and instead used exact and substring matching for coreference resolution. This showed a 5.7% decrease in the verb-only results. These results show that a protagonist greatly assists in narrative judgements.

## 5 Ordering Narrative Events

The model proposed in the previous section is designed to learn the major subevents in a narrative chain, but not how these events are ordered. In this section we extend the model to learn a partial temporal ordering of the events.

There are a number of algorithms for determining the temporal relationship between two events (Mani et al., 2006; Lapata and Lascarides, 2006; Chambers et al., 2007), many of them trained on the TimeBank Corpus (Pustejovsky et al., 2003) which codes events and their temporal relationships. The currently highest performing of these on raw data is the model of temporal labeling described in our previous work (Chambers et al., 2007). Other approaches have depended on hand tagged features.

Chambers et al. (2007) shows 59.4% accuracy on the classification task for six possible relations between pairs of events: *before*, *immediately-before*, *included-by*, *simultaneous*, *begins* and *ends*. We focus on the *before* relation because the others are less relevant to our immediate task. We combine *immediately-before* with *before*, and merge the other four relations into an *other* category. At the binary task of determining if one event is *before* or *other*, we achieve 72.1% accuracy on Timebank.

The above approach is a two-stage machine learning architecture. In the **first stage**, the model uses supervised machine learning to label temporal attributes of events, including tense, grammatical aspect, and aspectual class. This first stage classifier relies on features such as neighboring part of speech tags, neighboring auxiliaries and modals, and WordNet synsets. We use SVMs (Chambers et al. (2007) uses Naive Bayes) and see minor performance boosts on Timebank. These imperfect classifications, combined with other linguistic features, are then used in a **second stage** to classify the temporal relationship between two events. Other features include event-event syntactic properties such as the syntactic dominance relations between the two events, as well as new bigram features of tense, aspect and class (e.g. “present past” if the first event is in the present, and the second past), and whether the events occur in the same or different sentences.

## 5.1 Training a Temporal Classifier

We use the entire Timebank Corpus as supervised training data, condensing the *before* and *immediately-before* relations into one *before* relation. The remaining relations are merged into *other*.

The vast majority of potential event pairs in Timebank are unlabeled. These are often *none* relations (events that have no explicit relation) or as is of-

ten the case, *overlap* relations where the two events have no Timebank-defined ordering but overlap in time. Even worse, many events do have an ordering, but they were not tagged by the human annotators. This could be due to the overwhelming task of temporal annotation, or simply because some event orderings are deemed more important than others in understanding the document. We consider all untagged relations as *other*, and experiment with including none, half, and all of them in training.

Taking a cue from Mani et al. (2006), we also increased Timebank’s size by applying transitivity rules to the hand labeled data. The following is an example of the applied transitive rule:

**if** run BEFORE fall **and** fall BEFORE injured  
**then** run BEFORE injured

This increases the number of relations from 37519 to 45619. Perhaps more importantly for our task, of all the added relations, the *before* relation is added the most. We experimented with original vs. expanded Timebank and found the expanded performed slightly worse. The decline may be due to poor transitivity additions, as several Timebank documents contain inconsistent labelings. All reported results are from training *without* transitivity.

## 5.2 Temporal Classifier in Narrative Chains

We classify the Gigaword Corpus in two stages, once for the temporal features on each event (tense, grammatical aspect, aspectual class), and once between all pairs of events that share arguments. This allows us to classify the *before/other* relations between all potential narrative events.

The first stage is trained on Timebank, and the second is trained using the approach described above, varying the size of the *none* training relations. Each pair of events in a gigaword document that share a coreferring argument is treated as a separate ordering classification task. We count the resulting number of labeled *before* relations between each verb/dependency pair. Processing the entire corpus produces a database of event pair counts where confidence of two generic events A and B can be measured by comparing how many *before* labels have been seen versus their inverted order B and A<sup>5</sup>.

<sup>5</sup>Note that we train with the *before* relation, and so transposing two events is similar to classifying the *after* relation.

### 5.3 Temporal Evaluation

We want to evaluate temporal order at the narrative level, across all events within a chain. We envision narrative chains being used for tasks of coherence, among other things, and so it is desired to evaluate temporal decisions within a coherence framework. Along these lines, our test set uses actual narrative chains from documents, hand labeled for a partial ordering. We evaluate coherence of these true chains against a random ordering. The task is thus deciding which of the two chains is most coherent, the original or the random (baseline 50%)? We generated up to 300 random orderings for each test document, averaging the accuracy across all.

Our evaluation data is the same 69 documents used in the test set for learning narrative relations. The chain from each document is hand identified and labeled for a partial ordering using only the *before* relation. Ordering was done by the authors and all attempts were made to include every before relation that exists in the document, or that could be deduced through transitivity rules. Figure 4 shows an example and its full reversal, although the evaluation uses random orderings. Each edge is a distinct before relation and is used in the judgement score.

The coherence score for a partially ordered narrative chain is the sum of all the relations that our classified corpus agrees with, weighted by how certain we are. If the gigaword classifications disagree, a weighted negative score is given. Confidence is based on a logarithm scale of the difference between the counts of before and after classifications. Formally, the score is calculated as the following:

$$\sum_{E:x,y} \begin{cases} \log(D(x,y)) & \text{if } x\beta y \text{ and } B(x,y) > B(y,x) \\ -\log(D(x,y)) & \text{if } x\beta y \text{ and } B(y,x) > B(x,y) \\ -\log(D(x,y)) & \text{if } !x\beta y \text{ \& \!}y\beta x \text{ \& } D(x,y) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where  $E$  is the set of all event pairs,  $B(i, j)$  is how many times we classified events  $i$  and  $j$  as *before* in Gigaword, and  $D(i, j) = |B(i, j) - B(j, i)|$ . The relation  $i\beta j$  indicates that  $i$  is temporally before  $j$ .

### 5.4 Results

Our approach gives higher scores to orders that coincide with the pairwise orderings classified in our gigaword training data. The results are shown in figure 5. Of the 69 chains, 6 did not have any ordered events and were removed from the evaluation. We

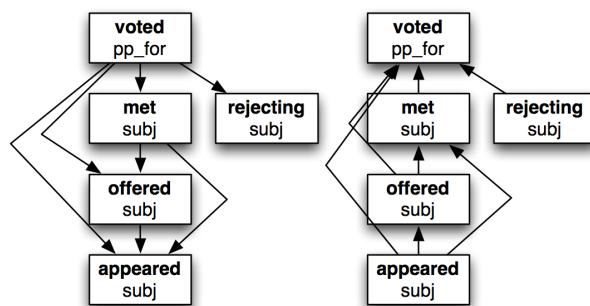


Figure 4: A narrative chain and its reverse order.

	All	$\geq 6$	$\geq 10$
correct	8086 <b>75%</b>	7603 <b>78%</b>	6307 <b>89%</b>
incorrect	1738	1493	619
tie	931	627	160

Figure 5: Results for choosing the correct ordered chain. ( $\geq 10$ ) means there were at least 10 pairs of ordered events in the chain.

generated (up to) 300 random orderings for each of the remaining 63. We report 75.2% accuracy, but 22 of the 63 had 5 or fewer pairs of ordered events. Figure 5 therefore shows results from chains with more than 5 pairs, and also 10 or more. As we would hope, the accuracy improves the larger the ordered narrative chain. We achieve 89.0% accuracy on the 24 documents whose chains most progress through time, rather than chains that are difficult to order with just the *before* relation.

Training without *none* relations resulted in high recall for *before* decisions. Perhaps due to data sparsity, this produces our best results as reported above.

## 6 Discrete Narrative Event Chains

Up till this point, we have learned narrative relations across all possible events, including their temporal order. However, the discrete lists of events for which Schank scripts are most famous have not yet been constructed.

We intentionally did not set out to reproduce explicit self-contained *scripts* in the sense that the ‘restaurant script’ is complete and cannot include other events. The name *narrative* was chosen to imply a *likely order* of events that is common in spoken and written retelling of world events. Discrete sets have the drawback of shutting out unseen and un-

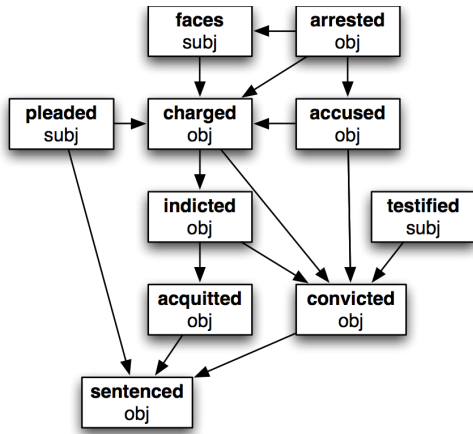


Figure 6: An automatically learned Prosecution Chain. Arrows indicate the *before* relation.

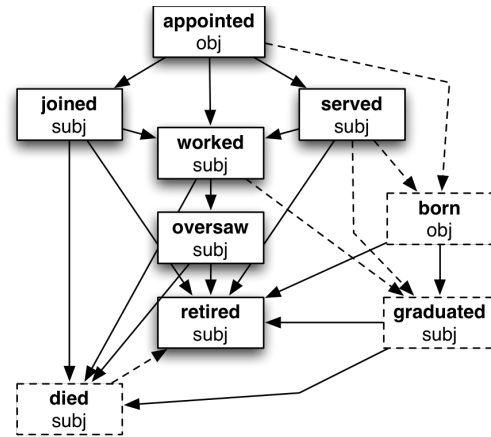


Figure 7: An Employment Chain. Dotted lines indicate incorrect *before* relations.

likely events from consideration. It is advantageous to consider a space of possible narrative events and the ordering within, not a closed list.

However, it is worthwhile to construct discrete narrative chains, if only to see whether the combination of event learning and ordering produce script-like structures. This is easily achievable by using the PMI scores from section 4 in an agglomerative clustering algorithm, and then applying the ordering relations from section 5 to produce a directed graph.

Figures 6 and 7 show two learned chains after clustering and ordering. Each arrow indicates a before relation. Duplicate arrows implied by rules of transitivity are removed. Figure 6 is remarkably accurate, and figure 7 addresses one of the chains from our introduction, the employment narrative. The core employment events are accurate, but clustering included life events (born, died, graduated) from obituaries of which some temporal information is incorrect. The Timebank corpus does not include obituaries, thus we suffer from sparsity in training data.

## 7 Discussion

We have shown that it is possible to learn narrative event chains unsupervised from raw text. Not only do our narrative relations show improvements over a baseline, but narrative chains offer hope for many other areas of NLP. Inference, coherence in summarization and generation, slot filling for question answering, and frame induction are all potential areas.

We learned a new measure of similarity, the nar-

rative relation, using the protagonist as a hook to extract a list of related events from each document. The 37% improvement over a verb-only baseline shows that we may not need presorted topics of documents to learn inferences. In addition, we applied state of the art temporal classification to show that sets of events can be partially ordered. Judgements of coherence can then be made over chains within documents. Further work in temporal classification may increase accuracy even further.

Finally, we showed how the event space of narrative relations can be clustered to create discrete sets. While it is unclear if these are better than an unconstrained distribution of events, they do offer insight into the quality of narratives.

An important area not discussed in this paper is the possibility of using narrative chains for semantic role learning. A narrative chain can be viewed as defining the semantic roles of an event, constraining it against roles of the other events in the chain. An argument's class can then be defined as the set of narrative arguments in which it appears.

We believe our model provides an important first step toward learning the rich causal, temporal and inferential structure of scripts and frames.

**Acknowledgment:** This work is funded in part by DARPA through IBM and by the DTO Phase III Program for AQUAINT through Broad Agency Announcement (BAA) N61339-06-R-0034. Thanks to the reviewers for helpful comments and the suggestion for a non-full-coreference baseline.



## References

- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In Christian Boitet and Pete Whitelock, editors, *ACL-98*, pages 86–90, San Francisco, California. Morgan Kaufmann Publishers.
- Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: an entity-based approach. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 141–148.
- David Bean and Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. *Proc. of HLT/NAACL*, pages 297–304.
- Samuel Brody. 2007. Clustering Clauses for High-Level Relation Detection: An Information-theoretic Approach. *Proceedings of the 43rd Annual Meeting of the Association of Computational Linguistics*, pages 448–455.
- Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of ACL-07*, Prague, Czech Republic.
- Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of EMNLP-04*.
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-06*, pages 449–454.
- Tony Deyes. 1984. Towards an authentic 'discourse cloze'. *Applied Linguistics*, 5(2).
- Toshiaki Fujiki, Hidetsugu Nanba, and Manabu Okumura. 2003. Automatic acquisition of script knowledge from a text collection. In *EACL*, pages 91–94.
- David Graff. 2002. English Gigaword. *Linguistic Data Consortium*.
- Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2).
- Mirella Lapata and Alex Lascarides. 2006. Learning sentence-internal temporal relations. In *Journal of AI Research*, volume 27, pages 85–117.
- C.Y. Lin and E. Hovy. 2000. The automated acquisition of topic signatures for text summarization. *Proceedings of the 17th conference on Computational linguistics-Volume 1*, pages 495–501.
- Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of ACL-06*, July.
- Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 681–687.
- Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17:21–43.
- Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. *Proceedings of HLT/NAACL*, 4:321–328.
- James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, David Day, Lisa Ferro, Robert Gaizauskas, Marcia Lazo, Andrea Setzer, and Beth Sundheim. 2003. The timebank corpus. *Corpus Linguistics*, pages 647–656.
- Roger C. Schank and Robert P. Abelson. 1977. *Scripts, plans, goals and understanding*. Lawrence Erlbaum.
- Wilson L. Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism Quarterly*, 30:415–433.

# Semantic Role Labeling Systems for Arabic using Kernel Methods

**Mona Diab**

CCLS, Columbia University  
New York, NY 10115, USA  
mdiab@ccls.columbia.edu

**Alessandro Moschitti**

DISI, University of Trento  
Trento, I-38100, Italy  
moschitti@disi.unitn.it

**Daniele Pighin**

FBK-irst; DISI, University of Trento  
Trento, I-38100, Italy  
pighin@fbk.eu

## Abstract

There is a widely held belief in the natural language and computational linguistics communities that Semantic Role Labeling (SRL) is a significant step toward improving important applications, e.g. question answering and information extraction. In this paper, we present an SRL system for Modern Standard Arabic that exploits many aspects of the rich morphological features of the language. The experiments on the pilot Arabic Propbank data show that our system based on Support Vector Machines and Kernel Methods yields a global SRL  $F_1$  score of 82.17%, which improves the current state-of-the-art in Arabic SRL.

## 1 Introduction

Shallow approaches to semantic processing are making large strides in the direction of efficiently and effectively deriving tacit semantic information from text. Semantic Role Labeling (SRL) is one such approach. With the advent of faster and more powerful computers, more effective machine learning algorithms, and importantly, large data resources annotated with relevant levels of semantic information, such as the FrameNet (Baker et al., 1998) and PropBank (Kingsbury and Palmer, 2003), we are seeing a surge in efficient approaches to SRL (Carreras and Màrquez, 2005).

SRL is the process by which predicates and their arguments are identified and their roles are defined in a sentence. For example, in the English sentence, ‘John likes apples.’, the predicate is ‘likes’ whereas ‘John’ and ‘apples’, bear the semantic role labels *agent* (ARG0) and *theme* (ARG1). The crucial fact about semantic roles is that regardless of the overt syntactic structure variation, the underlying predicates remain the same. Hence, for the sentence ‘John opened the door’ and ‘the door opened’, though ‘the door’ is the object of the first sentence

and the subject of the second, it is the ‘theme’ in both sentences. Same idea applies to passive constructions, for example.

There is a widely held belief in the NLP and computational linguistics communities that identifying and defining roles of predicate arguments in a sentence has a lot of potential for and is a significant step toward improving important applications such as document retrieval, machine translation, question answering and information extraction (Moschitti et al., 2007).

To date, most of the reported SRL systems are for English, and most of the data resources exist for English. We do see some headway for other languages such as German and Chinese (Erk and Pado, 2006; Sun and Jurafsky, 2004). The systems for the other languages follow the successful models devised for English, e.g. (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Chen and Rambow, 2003; Thompson et al., 2003; Pradhan et al., 2003; Moschitti, 2004; Xue and Palmer, 2004; Haghighi et al., 2005). In the same spirit and facilitated by the release of the SemEval 2007 Task 18 data<sup>1</sup>, based on the Pilot Arabic Propbank, a preliminary SRL system exists for Arabic<sup>2</sup> (Diab and Moschitti, 2007; Diab et al., 2007a). However, it did not exploit some special characteristics of the Arabic language on the SRL task.

In this paper, we present an SRL system for MSA that exploits many aspects of the rich morphological features of the language. It is based on a supervised model that uses support vector machines (SVM) technology (Vapnik, 1998) for argument boundary detection and argument classification. It is trained and tested using the pilot Arabic Propbank data released as part of the SemEval 2007 data. Given the lack of a reliable Arabic deep syntactic parser, we

<sup>1</sup><http://nlp.cs.swarthmore.edu/semeval/>

<sup>2</sup>We use Arabic to refer to Modern Standard Arabic (MSA).

use gold standard trees from the Arabic Tree Bank (ATB) (Maamouri et al., 2004).

This paper is laid out as follows: Section 2 presents facts about the Arabic language especially in relevant contrast to English; Section 3 presents the approach and system adopted for this work; Section 4 presents the experimental setup, results and discussion. Finally, Section 5 draws our conclusions.

## 2 Arabic Language and Impact on SRL

Arabic is a very different language from English in several respects relevant to the SRL task. Arabic is a semitic language. It is known for its templatic morphology where words are made up of roots and affixes. Clitics agglutinate to words. Clitics include prepositions, conjunctions, and pronouns.

In contrast to English, Arabic exhibits rich morphology. Similar to English, Arabic verbs explicitly encode tense, voice, Number, and Person features. Additionally, Arabic encodes verbs with Gender, Mood (subjunctive, indicative and jussive) information. For nominals (nouns, adjectives, proper names), Arabic encodes syntactic Case (accusative, genitive and nominative), Number, Gender and Definiteness features. In general, many of the morphological features of the language are expressed via short vowels also known as diacritics<sup>3</sup>.

Unlike English, syntactically Arabic is a pro-drop language, where the subject of a verb may be implicitly encoded in the verb morphology. Hence, we observe sentences such as *اكل البرتقال Akl AlbrtqAl* ‘ate-[he] the-oranges’, where the verb *Akl* encodes the third Person Masculine Singular subject in the verbal morphology. It is worth noting that in the ATB 35% of all sentences are pro-dropped for subject (Maamouri et al., 2006). Unless the syntactic parse is very accurate in identifying the pro-dropped case, identifying the syntactic subject and the underlying semantic arguments are a challenge for such pro-drop cases.

Arabic syntax exhibits relative free word order. Arabic allows for both subject-verb-object (SVO) and verb-subject-object (VSO) argument orders.<sup>4</sup> In

the VSO constructions, the verb agrees with the syntactic subject in Gender only, while in the SVO constructions, the verb agrees with the subject in both Number and Gender. Even though, in the ATB, an equal distribution of both VSO and SVO is observed (each appearing 30% of the time), it is known that in general Arabic is predominantly in VSO order. Moreover, the pro-drop cases could effectively be perceived as VSO orders for the purposes of SRL. Syntactic Case is very important in the cases of VSO and pro-drop constructions as they indicate the syntactic roles of the object arguments with accusative Case. Unless the morphology of syntactic Case is explicitly present, such free word order could run the SRL system into significant confusion for many of the predicates where both arguments are semantically of the same type.

Arabic exhibits more complex noun phrases than English mainly to express possession. These constructions are known as *idafa* constructions. Modern standard Arabic does not have a special particle expressing possession. In these complex structures a surface indefinite noun (missing an explicit definite article) may be followed by a definite noun marked with genitive Case, rendering the first noun syntactically definite. For example, *رجل البيت rjl Albyt* ‘man the-house’ meaning ‘man of the house’, *رجل* becomes definite. An adjective modifying the noun *رجل* will have to agree with it in Number, Gender, Definiteness, and Case. However, without explicit morphological encoding of these agreements, the scope of the arguments would be confusing to an SRL system. In a sentence such as *رجل البيت الطويل rjlu Albyti AlTwyly* meaning ‘the tall man of the house’: ‘man’ is definite, masculine, singular, nominative, corresponding to Definiteness, Gender, Number and Case, respectively; ‘the-house’ is definite, masculine, singular, genitive; ‘the-tall’ is definite, masculine, singular, nominative. We note that ‘man’ and ‘tall’ agree in Number, Gender, Case and Definiteness. Syntactic Case is marked using short vowels *u*, and *i* at the end of the word. Hence, *rjlu* and *AlTwyly* agree in their Case ending<sup>5</sup> Without the explicit marking of the Case information,

<sup>3</sup>Diacritics encode the vocalic structure, namely the short vowels, as well as the gemination marker for consonantal doubling, among other markers.

<sup>4</sup>MSA less often allows for OSV, or OVS.

<sup>5</sup>The presence of the *Albyti* is crucial as it renders *rjlu* definite therefore allowing the agreement with *AlTwyly* to be complete.

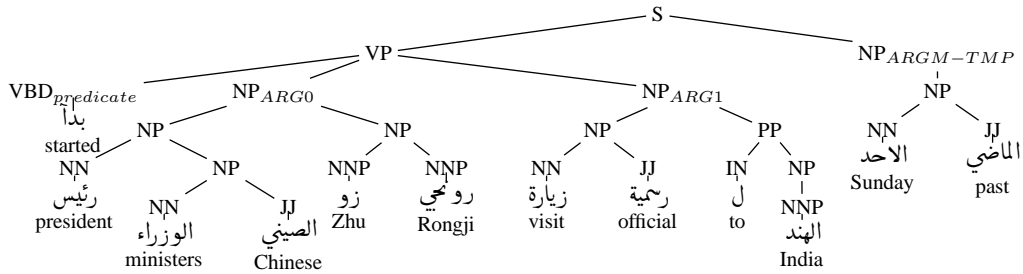


Figure 1: Annotated Arabic Tree corresponding to ‘Chinese Prime minister Zhu Rongji started an official visit to India last Sunday.’

namely in the word endings, it could be equally valid that ‘the-tall’ modifies ‘the-house’ since they agree in Number, Gender and Definiteness as explicitly marked by the Definiteness article *Al*. Hence, these *idafa* constructions could be tricky for SRL in the absence of explicit morphological features. This is compounded by the general absence of short vowels, expressed by diacritics (i.e. the *u* and *i* in *rjlu* and *Al-byti*.) in naturally occurring text. *Idafa* constructions in the ATB exhibit recursive structure, embedding other NPs, compared to English where possession is annotated with flat NPs and is designated by a possessive marker.

Arabic texts are underspecified for diacritics to different degrees depending on the genre of the text (Diab et al., 2007b). Such an underspecification of diacritics masks some of the very relevant morpho-syntactic interactions between the different categories such as agreement between nominals and their modifiers as exemplified before, or verbs and their subjects.

Having highlighted the differences, we hypothesize that the interaction between the rich morphology (if explicitly marked and present) and syntax could help with the SRL task. The presence of explicit Number and Gender agreement as well as Case information aids with identification of the syntactic subject and object even if the word order is relatively free. Gender, Number, Definiteness and Case agreement between nouns and their modifiers and other nominals, should give clues to the scope of arguments as well as their classes. The presence of such morpho-syntactic information should lead to better argument boundary detection and better classification.

### 3 An SRL system for Arabic

The previous section suggests that an optimal model should take into account specific characteristics of

Feature Name	Description
Predicate	Lemmatization of the predicate word
Path	Syntactic path linking the predicate and an argument, e.g. NN↑NP↑VP↓VBX
Partial path	<i>Path</i> feature limited to the branching of the argument
No-direction path	Like <i>Path</i> without traversal directions
Phrase type	Syntactic type of the argument node
Position	Relative position of the argument with respect to the predicate
Verb subcategorization	Production rule expanding the predicate parent node
Syntactic Frame	Position of the NPs surrounding the predicate
First and last word/POS	First and last words and POS tags of candidate argument phrases

Table 1: Standard linguistic features employed by most SRL systems.

Arabic. In this research, we go beyond the previously proposed basic SRL system for Arabic (Diab et al., 2007a; Diab and Moschitti, 2007). We exploit the full morphological potential of the language to verify our hypothesis that taking advantage of the interaction between morphology and syntax can improve on a basic SRL system for morphologically rich languages.

Similar to the previous Arabic SRL systems, our adopted SRL models use Support Vector Machines to implement a two step classification approach, i.e. boundary detection and argument classification. Such models have already been investigated in (Pradhan et al., 2005; Moschitti et al., 2005). The two step classification description is as follows.

#### 3.1 Predicate Argument Extraction

The extraction of predicative structures is based on the sentence level. Given a sentence, its predicates, as indicated by verbs, have to be identified along with their arguments. This problem is usually divided in two subtasks: (a) the detection of the target argument boundaries, i.e. the span of the argument words in the sentence, and (b) the classification of the argument type, e.g. *Arg0* or *ArgM* for Propbank

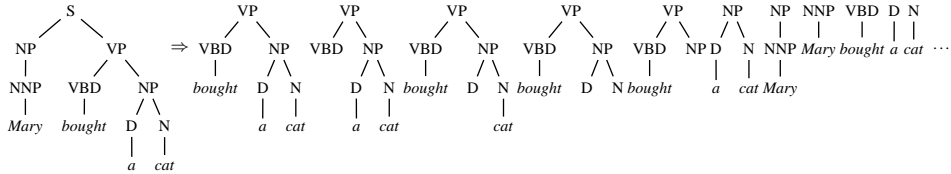


Figure 2: Fragment space generated by a tree kernel function for the sentence *Mary bought a cat*.

or *Agent* and *Goal* for the FrameNet.

The standard approach to learn both the detection and the classification of predicate arguments is summarized by the following steps:

- (a) Given a sentence from the *training-set*, generate a full syntactic parse-tree;
- (b) let  $\mathcal{P}$  and  $\mathcal{A}$  be the set of predicates and the set of parse-tree nodes (i.e. the potential arguments), respectively;
- (c) for each pair  $\langle p, a \rangle \in \mathcal{P} \times \mathcal{A}$ : extract the feature representation set,  $F_{p,a}$  and put it in  $T^+$  (positive examples) if the subtree rooted in  $a$  covers exactly the words of one argument of  $p$ , otherwise put it in  $T^-$  (negative examples).

For instance, in Figure 1, for each combination of the predicate *started* with the nodes NP, S, VP, VPD, NNP, NN, PP, JJ or IN the instances  $F_{started,a}$  are generated. In case the node  $a$  exactly covers ‘president ministers Chinese Zhu Rongji’ or ‘visit official to India’,  $F_{p,a}$  will be a positive instance otherwise it will be a negative one, e.g.  $F_{started,IN}$ .

The  $T^+$  and  $T^-$  sets are used to train the boundary classifier. To train the multi-class classifier,  $T^+$  can be reorganized as positive  $T_{arg_i}^+$  and negative  $T_{arg_i}^-$  examples for each argument  $i$ . This way, an individual ONE-vs-ALL classifier for each argument  $i$  can be trained. We adopt this solution, according to (Pradhan et al., 2005), since it is simple and effective. In the classification phase, given an unseen sentence, all its  $F_{p,a}$  are generated and classified by each individual classifier  $C_i$ . The argument associated with the maximum among the scores provided by the individual classifiers is eventually selected.

The above approach assigns labels independently, without considering the whole predicate argument structure. As a consequence, the classifier output may generate overlapping arguments. Thus, to make the annotations globally consistent, we apply a disambiguating heuristic adopted from (Diab and Moschitti, 2007) that selects only one argument among multiple overlapping arguments.

### 3.2 Features

The discovery of relevant features is, as usual, a complex task. The choice of features is further compounded for a language such as Arabic given its rich morphology and morpho-syntactic interactions.

To date, there is a common consensus on the set of basic standard features for SRL, which we will refer to as *standard*. The set of standard features, refers to unstructured information derived from parse trees. e.g. *Phrase Type*, *Predicate Word* or *Head Word*. Typically the standard features are language independent. In our experiments we employ the features listed in Table 1, defined in (Gildea and Jurafsky, 2002; Pradhan et al., 2005; Xue and Palmer, 2004). For example, the *Phrase Type* indicates the syntactic type of the phrase labeled as a predicate argument, e.g. NP for *ARG1* in Figure 1. The *Parse Tree Path* contains the path in the parse tree between the predicate and the argument phrase, expressed as a sequence of nonterminal labels linked by direction (up or down) symbols, e.g. VBD  $\uparrow$  VP  $\downarrow$  NP for *ARG1* in Figure 1. The *Predicate Word* is the surface form of the verbal predicate, e.g. *started* for all arguments. The standard features, as successful as they are, are designed primarily for English. They are not exploiting the different characteristics of the Arabic language as expressed through morphology. Hence, we explicitly encode new SRL features that capture the richness of Arabic morphology and its role in morpho-syntactic behavior. The set of morphological attributes include: inflectional morphology such as Number, Gender, Definiteness, Mood, Case, Person; derivational morphology such as the Lemma form of the words with all the diacritics explicitly marked; vowelized and fully diacritized form of the surface form; the English gloss<sup>6</sup>. It is worth noting that there exists highly accurate morphological taggers for Arabic such as the MADA system (Habash and Rambow, 2005; Roth et al., 2008). MADA tags

<sup>6</sup>The gloss is not sense disambiguated, hence they include homonyms.

Feature Name	Description
Definiteness	Applies to nominals, values are definite, indefinite or inapplicable
Number	Applies to nominals and verbs, values are singular, plural or dual or inapplicable
Gender	Applies to nominals, values are feminine, masculine or inapplicable
Case	Applies to nominals, values are accusative, genitive, nominative or inapplicable
Mood	Applies to verbs, values are subjunctive, indicative, jussive or inapplicable
Person	Applies to verbs and pronouns, values are 1st, 2nd, 3rd person or inapplicable
Lemma	The citation form of the word fully diacritized with the short vowels and gemination markers if applicable
Gloss	this is the corresponding English meaning as rendered by the underlying lexicon.
Vocalized word	The surface form of the word with all the relevant diacritics. Unlike Lemma, it includes all the inflections.
Unvowelized word	The naturally occurring form of the word in the sentence with no diacritics.

Table 2: Rich morphological features encoded in the Extended Argument Structure Tree (EAST).

modern standard Arabic with all the relevant morphological features as well as it produces highly accurate lemma and gloss information by tapping into an underlying morphological lexicon. A list of the extended features is described in Table 2.

The set of possible features and their combinations are very large leading to an intractable feature selection problem. Therefore, we exploit well known kernel methods, namely tree kernels, to robustly experiment with all the features simultaneously. Such kernel engineering, as shown in (Moschitti, 2004), allows us to experiment with many syntactic/semantic features seamlessly.

### 3.3 Engineering Arabic Features with Kernel Methods

Feature engineering via kernel methods is a useful technique that allows us to save a lot of time in the design and implementation of features. The basic idea is (a) to design a set of basic value-attribute features and apply polynomial kernels and generate all possible combinations; or (b) to design basic tree structures expressing properties related to the target linguistic objects and use tree kernels to generate all possible tree subparts, which will constitute the feature representation vectors for the learning algorithm.

Tree kernels evaluate the similarity between two trees in terms of their overlap, generally measured as the number of common substructures (Collins and Duffy, 2002). For example, Figure 2, shows a small parse tree and some of its fragments. To design a function which computes the number of common substructures between two trees  $t_1$  and  $t_2$ , let us define the set of fragments  $\mathcal{F}=\{f_1, f_2, \dots\}$  and the indicator function  $I_i(n)$ , equal to 1 if the target  $f_i$  is rooted at node  $n$  and 0 otherwise. A tree kernel function  $K_T(\cdot)$  over two trees is defined as:

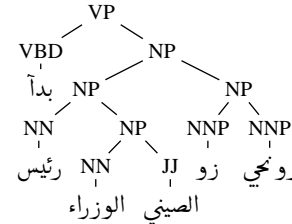


Figure 3: Example of the positive AST structured feature encoding the argument ARG0 in the sentence depicted in Figure 1.

$K_T(t_1, t_2) = \sum_{n_1 \in N_{t_1}} \sum_{n_2 \in N_{t_2}} \Delta(n_1, n_2)$ , where  $N_{t_1}$  and  $N_{t_2}$  are the sets of nodes of  $t_1$  and  $t_2$ , respectively. The function  $\Delta(\cdot)$  evaluates the number of common fragments rooted in  $n_1$  and  $n_2$ , i.e.  $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1)I_i(n_2)$ .  $\Delta$  can be efficiently computed with the algorithm proposed in (Collins and Duffy, 2002).

### 3.4 Structural Features for Arabic

In order to incorporate the characteristically rich Arabic morphology features structurally in the tree representations, we convert the features into value-attribute pairs at the leaf node level of the tree. Fig 1 illustrates the morphologically underspecified tree with some of the morphological features encoded in the POS tag such as VBD indicating past tense. This contrasts with Fig. 4 which shows an excerpt of the same tree encoding the chosen relevant morphological features.

For the sake of classification, we will be dealing with two kinds of structures: the Argument Structure Tree (AST) (Pighin and Basili, 2006) and the Extended Argument Structure Tree (EAST). The AST is defined as the minimal subtree encompassing all and only the leaf nodes encoding words belonging to the predicate or one of its arguments. An AST example is shown in Figure 3. The EAST is the corresponding structure in which all the leaf nodes have been extended with the ten morphological fea-

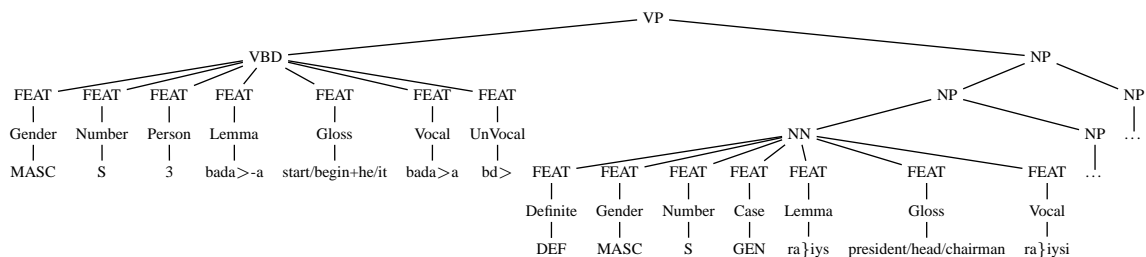


Figure 4: An excerpt of the EAST corresponding to the AST shown in Figure 3, with attribute-value extended morphological features represented as leaf nodes.

tures described in Table 2, forming a vector of 10 preterminal-terminal node pairs that replace the surface of the leaf. The resulting EAST structure is shown in Figure 4.

Not all the features are instantiated for all the leaf node words. Due to space limitations, in the figure we did not include the Features that have NULL values. For instance, Definiteness is always associated with nominals, hence the verb بدأ *bd* ‘start’ is assigned a NULL value for the Definite feature. Verbs exhibit Gender information depending on inflections. For our example, بدأ ‘started’ is inflected for masculine Gender, singular Number, third person. On the other hand, the noun الوزراء is definite and is assigned genitive Case since it is in a possessive, *idafa*, construction.

The features encoded by the EAST can provide very useful hints for boundary and role classification. Considering Figure 1, argument boundaries is not as straight forward to identify as there are several NPs. Assuming that the inner most NP ‘ministers the-Chinese’ is a valid Argument could potentially be accepted. There is ample evidence that any NN followed by a JJ would make a perfectly valid Argument. However, an AST structure would mask the fact that the JJ ‘the-Chinese’ does not modify the NN ‘ministers’ since they do not agree in Number<sup>7</sup>, and in syntactic Case, where the latter is genitive and the former is nominative. ‘the-Chinese’ in fact modifies ‘president’ as they agree on all the underlying morphological features. Conversely, the EAST in Figure 4 explicitly encodes this agreement including an agreement on Definiteness. It is worth noting that just observing the Arabic word رئيس ‘president’ in Fig 1, the system would assume that it is an indefinite word since it does not include the definite arti-

<sup>7</sup>The POS tag on this node is NN as broken plural, however, the underlying morphological feature Number is plural.

cle ال. Therefore, the system could be lead astray to conclude that ‘the-Chinese’ does not modify ‘president’ but rather ‘the-ministers’. Without knowing the Case information and the agreement features between the verb بدأ ‘started’ and the two nouns heading the two main NPs in our tree, the syntactic subject can be either زيارة ‘visit’ or رئيس ‘president’ in Figure 1. The EAST is more effective in identifying the first noun as the syntactic subject and the second as the object since the morphological information indicates that they are in nominative and accusative Case, respectively. Also the agreement in Gender and Number between the verb and the syntactic subject is identified in the enriched tree. We see that بدأ ‘started’ and رئيس ‘president’ agree in being singular and masculine. If زيارة ‘visit’ were the syntactic subject, we would have seen the verb inflected as بدأت ‘started-FEM’ with a feminine inflection to reflect the verb-subject agreement on Gender. Hence these agreement features should help with the classification task.

## 4 Experiments

In these experiments we investigate (a) if the technology proposed in previous work for automatic SRL of English texts is suitable for Arabic SRL systems, and (b) the impact of tree kernels using new tree structures on Arabic SRL. For this purpose, we test our models on the two individual phases of the traditional 2-stage SRL model (i.e. boundary detection and argument classification) and on the complete SRL task. We use three different feature spaces: a set of standard attribute-value features and the AST and the EAST structures defined in 3.4. Standard feature vectors can be combined with a polynomial kernel (Poly), which, when the degree is larger than 1, automatically generates feature conjunctions. This, as suggested in (Pradhan et al., 2005; Moschitti, 2004), can help stressing the differ-

ences between different argument types. Tree structures can be used in the learning algorithm thanks to the tree kernels described in Section 3.3. Moreover, to verify if the above feature sets are equivalent or complementary, we can join them by means of additive operation which always produces a valid kernel (Shawe-Taylor and Cristianini, 2004).

#### 4.1 Experimental setup

We use the dataset released in the SemEval 2007 Task 18 on Arabic Semantic Labeling (Diab et al., 2007a). The data covers the 95 most frequent verbs in the Arabic Treebank III ver. 2 (ATB). The ATB consists of MSA newswire data from the Annhar newspaper, spanning the months from July to November, 2002. All our experiments are carried out with gold standard trees.

An important characteristic of the dataset is the use of unvowelized Arabic in the Buckwalter transliteration scheme for deriving the basic features for the AST experimental condition. The data comprises a development set, a test set and a training set of 886, 902 and 8,402 sentences, respectively, where each set contain 1725, 1661 and 21,194 argument instances. These instances are distributed over 26 different role types. The training instances of the boundary detection task also include parse-tree nodes that do not correspond to correct boundaries (we only considered 350K examples). For the experiments, we use SVM-Light-TK toolkit<sup>8</sup> (Moschitti, 2004; Moschitti, 2006) and its SVM-Light default parameters. The system performance, i.e.  $F_1$  on single boundary and role classifier, accuracy of the role multi-classifier and the  $F_1$  of the complete SRL systems, are computed by means of the CoNLL evaluator<sup>9</sup>.

#### 4.2 Results

Figure 5 reports the  $F_1$  of the SVM boundary classifier using Polynomial Kernels with a degree from 1 to 6 (i.e. Poly $i$ ), the AST and the EAST kernels and their combinations. We note that as we introduce conjunctions, i.e. a degree larger than 2, the  $F_1$  increases by more than 3 percentage points. Thus, not only are the English features meaningful for Arabic but also their combinations are important, reveal-

<sup>8</sup><http://disi.unitn.it/~moschitti>

<sup>9</sup><http://www.lsi.upc.es/~srlconll/soft.html>

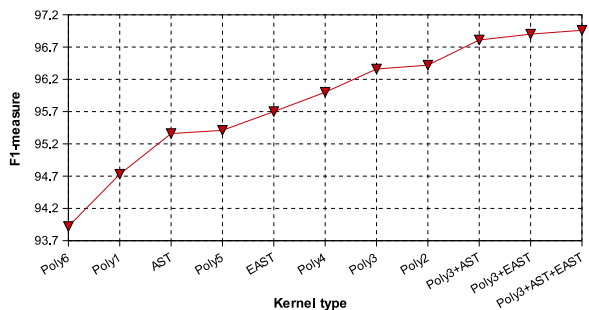


Figure 5: Impact of polynomial kernel, tree kernels and their combinations on boundary detection.

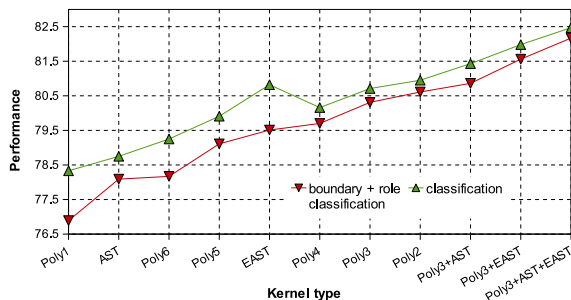


Figure 6: Impact of the polynomial kernel, tree kernels and their combinations on the accuracy in role classification (gold boundaries) and on the  $F_1$  of complete SRL task (boundary + role classification).

ing that both languages share an underlying syntax-semantic interface. Moreover, we note that the  $F_1$  of EAST is higher than the  $F_1$  of AST which in turn is higher than the linear kernel (Poly1). However, when conjunctive features (Poly2-4) are used the system accuracy exceeds those of tree kernel models alone. Further increasing the polynomial degree (Poly5-6) generates very complex hypotheses which result in very low accuracy values.

Therefore, to improve the polynomial kernel, we sum it to the contribution of AST and/or EAST, obtaining AST+Poly3 (polynomial kernel of degree 3), EAST+Poly3 and AST+EAST+Poly3, whose  $F_1$  scores are also shown in Figure 5. Such combined models improve on the best polynomial kernel. However, not much difference is shown between AST and EAST on boundary detection. This is expected since we are using gold standard trees. We hypothesize that the rich morphological features will help more with the role classification task. Therefore, we evaluate role classification with gold boundaries. The curve labeled "classification" in Figure 6 illustrates the accuracy of the SVM role multi-classifier according to different kernels.



	P3	AST	EAST	AST+ P3	EAST+ P3	AST+ EAST+ P3
P	81.73	80.33	81.7	81.73	82.46	83.08
R	78.93	75.98	77.42	80.01	80.67	81.28
F <sub>1</sub>	80.31	78.09	79.51	80.86	81.56	82.17

Table 3: F<sub>1</sub> of different models on the Arabic SRL task.

Again, we note that a degree larger than 1 yields a significant improvement of more than 3 percent points, suggesting that the design of Arabic SRL system based on SVMs requires polynomial kernels. In contrast to the boundary results, EAST highly improves over AST (by about 3 percentage points) and produces an F<sub>1</sub> comparable to the best Polynomial kernel. Moreover, AST+Poly3, EAST+Poly3 and AST+EAST+Poly3 all yield different degrees of improvement, where the latter model is both the richest in terms of features and the most accurate.

These results strongly suggest that: (a) tree kernels generate new syntactic features that are useful for the classification of Arabic semantic roles; (b) the richer morphology of Arabic language should be exploited effectively to obtain accurate SRL systems; (c) tree kernels appears to be a viable approach to effectively achieve this goal.

To illustrate the practical feasibility of our system, we investigate the complete SRL task where both the boundary detection and argument role classification are performed automatically. The curve labeled "boundary + role classification" in Figure 6 reports the F<sub>1</sub> of SRL systems based on the previous kernels. The trend of the plot is similar to the gold-standard boundaries case. The difference among the F<sub>1</sub> scores of the AST+Poly3, EAST+Poly3 and AST+EAST+Poly3 is slightly reduced. This may be attributed to the fact that they produce similar boundary detection results, which in turn, for the global SRL outcome, are summed to those of the classification phase. Table 3 details the differences among the models and shows that the best model improves the SRL system based on the polynomial kernel, i.e. the SRL state-of-the-art for Arabic, by about 2 percentage points. This is a very large improvement for SRL systems (Carreras and Màrquez, 2005). These results confirm that the new enriched structures along with tree kernels are a promising approach for Arabic SRL systems.

Finally, Table 4 reports the F<sub>1</sub> of the best model, AST+EAST+Poly3, for individual arguments in the

Role	Precision	Recall	F <sub>β=1</sub>
ARG0	96.14%	97.27%	96.70
ARG0-STR	100.00%	20.00%	33.33
ARG1	88.52%	92.70%	90.57
ARG1-STR	33.33%	15.38%	21.05
ARG2	69.35%	76.67%	72.82
ARG3	66.67%	16.67%	26.67
ARGM-ADV	66.98%	61.74%	64.25
ARGM-CAU	100.00%	9.09%	16.67
ARGM-CND	25.00%	33.33%	28.57
ARGM-LOC	67.44%	95.08%	78.91
ARGM-MNR	54.00%	49.09%	51.43
ARGM-NEG	80.85%	97.44%	88.37
ARGM-PRD	20.00%	8.33%	11.76
ARGM-PRP	85.71%	66.67%	75.00
ARGM-TMP	91.35%	88.79%	90.05

Table 4: SRL F<sub>1</sub> of the single arguments using the AST+EAST+Poly3 kernel.

SRL task. We note that, as for English SRL, ARG0 shows high values (96.70%). Conversely, ARG1 seems more difficult to be classified in Arabic. The F<sub>1</sub> for ARG1 is only 90.57% compared with 96.70% for ARG0.

This may be attributed to the different possible syntactic orders of Arabic constructions confusing the syntactic subject with the object especially where there is no clear morphological features on the arguments to decide either way.

## 5 Conclusions

We have presented a model for Arabic SRL that yields a global SRL F<sub>1</sub> score of 82.17% by combining rich structured features and traditional attribute-value features derived from English SRL systems. The resulting system significantly improves previously reported results on the same task and dataset. This outcome is very promising given that the available data is small compared to the English data sets.

For future work, we would like to explore further explicit morphological features such as aspect tense and voice as well as richer POS tag sets such as those proposed in (Diab, 2007). Finally, we would like to experiment with automatic parses and different syntactic formalisms such as dependencies and shallow parses.

## Acknowledgements

Mona Diab is partly funded by DARPA Contract No. HR0011-06-C-0023. Alessandro Moschitti has been partially funded by CCLS of the Columbia University and by the FP6 IST LUNA project contract no 33549.

## References

- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *COLING-ACL '98: University of Montréal*.
- Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*, Ann Arbor, Michigan.
- John Chen and Owen Rambow. 2003. Use of Deep Linguistic Features for the Recognition and Labeling of Semantic Arguments. In *Proceedings of EMNLP*, Sapporo, Japan.
- Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete structures, and the voted perceptron. In *ACL02*.
- Mona Diab and Alessandro Moschitti. 2007. Semantic Parsing for Modern Standard Arabic. In *Proceedings of RANLP*, Borovets, Bulgaria.
- Mona Diab, Musa Alkhalifa, Sabry ElKateb, Christiane Fellbaum, Aous Mansouri, and Martha Palmer. 2007a. Semeval-2007 task 18: Arabic Semantic Labeling. In *Proceedings of SemEval-2007*, Prague, Czech Republic.
- Mona Diab, Mahmoud Ghoneim, and Nizar Habash. 2007b. Arabic Diacritization in the Context of Statistical Machine Translation. In *Proceedings of MT-Summit*, Copenhagen, Denmark.
- Mona Diab. 2007. Towards an Optimal Pos Tag Set for Modern Standard Arabic Processing. In *Proceedings of RANLP*, Borovets, Bulgaria.
- Katrin Erk and Sebastian Pado. 2006. Shalmaneser – A Toolchain for Shallow Semantic Parsing. *Proceedings of LREC*.
- Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*.
- Daniel Gildea and Martha Palmer. 2002. The Necessity of Parsing for Predicate Argument Recognition. In *Proceedings of ACL-02*, Philadelphia, PA, USA.
- Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of ACL'05*, Ann Arbor, Michigan.
- Aria Haghighi, Kristina Toutanova, and Christopher Manning. 2005. A Joint Model for Semantic Role Labeling. In *Proceedings of CoNLL-2005*, Ann Arbor, Michigan.
- Paul Kingsbury and Martha Palmer. 2003. Propbank: the Next Level of Treebank. In *Proceedings of Treebanks and Lexical Theories*.
- Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank : Building a Large-Scale Annotated Arabic Corpus.
- Mohamed Maamouri, Ann Bies, Tim Buckwalter, Mona Diab, Nizar Habash, Owen Rambow, and Dalila Tabessi. 2006. Developing and Using a Pilot Dialectal Arabic Treebank.
- Alessandro Moschitti, Ana-Maria Giuglea, Bonaventura Coppola, and Roberto Basili. 2005. Hierarchical Semantic Role Labeling. In *Proceedings of CoNLL-2005*, Ann Arbor, Michigan.
- Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting Syntactic and Shallow Semantic Kernels for Question Answer Classification. In *Proceedings of ACL'07*, Prague, Czech Republic.
- Alessandro Moschitti. 2004. A Study on Convolution Kernels for Shallow Semantic Parsing. In *proceedings of ACL'04*, Barcelona, Spain.
- Alessandro Moschitti. 2006. Making Tree Kernels Practical for Natural Language Learning. In *Proceedings of EACL'06*.
- Alessandro Moschitti, Daniele Pighin and Roberto Basili. 2006. Semantic Role Labeling via Tree Kernel Joint Inference. In *Proceedings of CoNLL-X*.
- Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2003. Semantic Role Parsing: Adding Semantic Structure to Unstructured Text. In *Proceedings ICDM'03*, Melbourne, USA.
- Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support Vector Learning for Semantic Argument Classification. *Machine Learning*.
- Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking. In *ACL'08, Short Papers*, Columbus, Ohio, June.
- John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.
- Honglin Sun and Daniel Jurafsky. 2004. Shallow Semantic Parsing of Chinese. In *Proceedings of NAACL-HLT*.
- Cynthia A. Thompson, Roger Levy, and Christopher Manning. 2003. A Generative Model for Semantic Role Labeling. In *ECML'03*.
- Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley and Sons.
- Nianwen Xue and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, Barcelona, Spain.

# An Unsupervised Approach to Biography Production using Wikipedia

Fadi Biadsy,<sup>†</sup> Julia Hirschberg<sup>†</sup> and Elena Filatova<sup>\*</sup>

<sup>†</sup>Department of Computer Science  
Columbia University, New York, NY 10027, USA  
{fadi, julia}@cs.columbia.edu

<sup>\*</sup>InforSense LLC  
Cambridge, MA 02141, USA  
efilatova@inforsense.com

## Abstract

We describe an unsupervised approach to multi-document sentence-extraction based summarization for the task of producing biographies. We utilize Wikipedia to automatically construct a corpus of biographical sentences and TDT4 to construct a corpus of non-biographical sentences. We build a biographical-sentence classifier from these corpora and an SVM regression model for sentence ordering from the Wikipedia corpus. We evaluate our work on the DUC2004 evaluation data and with human judges. Overall, our system significantly outperforms all systems that participated in DUC2004, according to the ROUGE-L metric, and is preferred by human subjects.

## 1 Introduction

Producing biographies by hand is a labor-intensive task, generally done only for famous individuals. The process is particularly difficult when persons of interest are **not** well known and when information must be gathered from a wide variety of sources. We present an automatic, unsupervised, multi-document summarization (MDS) approach based on extractive techniques to producing biographies, answering the question “Who is X?”

There is growing interest in automatic MDS in general due in part to the explosion of multilingual and multimedia data available online. The goal of MDS is to automatically produce a concise, well-organized, and fluent summary of a set of documents on the same topic. MDS strategies have been

employed to produce both generic summaries and query-focused summaries. Due to the complexity of text generation, most summarization systems employ sentence-extraction techniques, in which the most relevant sentences from one or more documents are selected to represent the summary. This approach is guaranteed to produce grammatical sentences, although they must subsequently be ordered appropriately to produce a coherent summary.

In this paper we describe a sentence-extraction based MDS procedure to produce biographies from online resources automatically. We make use of Wikipedia, the largest free multilingual encyclopedia on the internet, to build a biographical-sentence classifier and a component for ordering sentences in the output summary. Section 2 presents an overview of our system. In Section 3 we describe our corpus and in Section 4 we discuss the components of our system in more detail. In Section 5, we present an evaluation of our work on the Document Understanding Conference of 2004 (DUC2004), the biography task (task 5) test set. In Section 6 we compare our research with previous work on biography generation. We conclude in Section 7 and identify directions for future research.

## 2 System Overview

In this section, we present an overview of our biography extraction system. We assume as input a set of documents retrieved by an information retrieval engine from a query consisting of the name of the person for whom the biography is desired. We further assume that these documents have been tagged with Named Entities (NE)s with coreferences resolved

using a system such as NYU’s 2005 ACE system (Grishman et al., 2005), which we used for our experiments. Our task is to produce a concise biography from these documents.

First, we need to select the most ‘important’ biographical sentences for the target person. To do so, we first extract from the input documents all sentences that contain some reference to the target person according to the coreference assignment algorithm; this reference may be the target’s name or a coreferential full NP or pronominal referring expression, such as *the President* or *he*. We call these sentences *hypothesis sentences*. We hypothesize that most ‘biographical’ sentences will contain a reference to the target. However, some of these sentences may be irrelevant to a biography; therefore, we filter them using a binary classifier that retains only ‘biographical’ sentences. These biographical sentences may also include redundant information; therefore, we cluster them and choose one sentence from each cluster to represent the information in that cluster. Since some of these sentences have more salient biographical information than others and since manually produced biographies tend to include information in a certain order, we reorder our summary sentences using an SVM regression model trained on biographies. Finally, the first reference to the target person in the initial sentence in the reordering is rewritten using the longest coreference in our hypothesis sentences which contains the target’s full name. We then trim the output to a threshold to produce a biography of a certain length for evaluation against the DUC2004 systems.

### 3 Training Data

One of the difficulties inherent in automatic biography generation is the lack of training data. One might collect training data by manually annotating a suitable corpus containing biographical and non-biographical data about a person, as in (Zhou et al., 2004). However, such annotation is labor intensive. To avoid this problem, we adopt an unsupervised approach. We use Wikipedia biographies as our corpus of ‘biographical’ sentences. We collect our ‘non-biographical’ sentences from the English newswire documents in the TDT4 corpus.<sup>1</sup> While each corpus

<sup>1</sup><http://projects.ldc.upenn.edu/TDT4>

may contain positive and negative examples, we assume that most sentences in Wikipedia biographies are biographical and that the majority of TDT4 sentences are non-biographical.

#### 3.1 Constructing the Biographical Corpus

To automatically collect our biographical sentences, we first download the xml version of Wikipedia and extract only the documents whose authors used the Wikipedia biography template when creating their biography. There are 16,906 biographies in Wikipedia that used this template. We next apply simple text processing techniques to clean the text. We select at most the first 150 sentences from each page, to avoid sentences that are not critically important to the biography. For each of these sentences we perform the following steps:

1. We identify the biography’s subject from its title, terming this name the ‘target person.’
2. We run NYU’s 2005 ACE system (Grishman et al., 2005) to tag NEs and do coreference resolution. There are 43 unique NE tags in our corpora, including *PER\_Individual*, *ORG\_Educational*, and so on, and TIMEX tags for all dates.
3. For each sentence, we replace each NE by its tag name and type ([name-type\_subtype]) as assigned by the NYU tagger. This modified sentence we term a *class-based/lexical sentence*.
4. Each non-pronominal referring expression (e.g., *George W. Bush, the US president*) that is tagged as coreferential with the target person is replaced by our own [TARGET\_PER] tag and every pronoun *P* that refers to the target person is replaced by [TARGET\_P], where *P* is the pronoun itself. This allows us to generalize our sentences while retaining a) the essential distinction between this NE (and its role in the sentence) and all other NEs in the sentence, and b) the form of referring expressions.
5. Sentences containing no reference to the target person are assumed to be irrelevant and removed from the corpus, as are sentences with

fewer than 4 tokens; short sentences are unlikely to contain useful information beyond the target reference.

For example, given sentences from the Wikipedia biography of Martin Luther King, Jr. we produce class-based/lexical sentences as follows:

Martin Luther King, Jr., was born on January 15, 1929, in Atlanta, Georgia. He was the son of Reverend Martin Luther King, Sr. and Alberta Williams King. He had an older sister, Willie Christine (September 11, 1927) and a younger brother, Albert Daniel.

[TARGET\_PER], was born on [TIMEX], in [GPE.PopulationCenter]. [TARGET\_HE] was the son of [PER\_Individual] and [PER\_Individual]. [TARGET\_HE] had an older sister, [PER\_Individual] ([TIMEX]) and a younger brother, [PER\_Individual].

### 3.2 Constructing the Non-Biographical Corpus

We use the TDT4 corpus to identify non-biographical sentences. Again, we run NYU’s 2005 ACE system to tag NEs and do coreference resolution on each news story in TDT4. Since we have no target name for these stories, we select an NE tagged as *PER\_Individual* at random from all NEs in the story to represent the target person. We exclude any sentence with no reference to this target person and produce class-based/lexical sentences as above.

## 4 Our Biography Extraction System

### 4.1 Classifying Biographical Sentences

Using the biographical and non-biographical corpora described in Section 3, we train a binary classifier to determine whether a new sentence should be included in a biography or not. For our experiments we extracted 30,002 sentences from Wikipedia biographies and held out 2,108 sentences for testing. Similarly, we extracted 23,424 sentences from TDT4, and held out 2,108 sentences for testing. For each sentence, we then extract the frequency of three class-based/lexical features — unigram, bigram, and trigram — and two POS features — the frequency of unigram and bigram POS. To reduce the dimensionality of our feature space, we first sort the features in decreasing order of Chi-square statistics computed from the contingency tables of the observed frequencies from the training data. We then take the highest 30-80% features, where the number of features used is determined empirically for

<i>Classifier</i>	<i>Accuracy</i>	<i>F-Measure</i>
<i>SVM</i>	87.6%	0.87
<i>M. naïve Bayes</i>	84.1%	0.84
<i>C4.5</i>	81.8%	0.82

Table 1: Binary classification results: Wikipedia biography class-based/lexical sentences vs. TDT4 class-based/lexical sentences

each feature type. This process identifies features that significantly contribute to the classification task. We extract 3K class-based/lexical unigrams, 5.5K bigrams, 3K trigrams, 20 POS unigrams, and 166 POS bigrams.

Using the training data described above, we experimented with three different classification algorithms using the Weka machine learning toolkit (Witten et al., 1999): multinomial naïve Bayes, SVM with linear kernel, and C4.5. Weka also provides a classification confidence score that represents how confident the classifier is on each classified sample, which we will make use of as well.

Table 1 presents the classification results on our 4,216 held-out test-set sentences. These results are quite promising. However, we should note that they may not necessarily represent the successful classification of biographical vs. non-biographical sentences but rather the classification of Wikipedia sentences vs. TDT4 sentences. We will validate these results for our full systems in Section 5.

### 4.2 Removing Redundant Sentences

Typically, redundancy removal is a standard component in MDS systems. In sentence-extraction based summarizers, redundant sentences are defined as those which include the same information without introducing new information and identified by some form of lexically-based clustering. We use an implementation of a single-link nearest neighbor clustering technique based on stem-overlap (Blair-Goldensohn et al., 2004b) to cluster the sentences classified as biographical by our classifier, and then select the sentence from each cluster that maximizes the confidence score returned by the classifier as the representative for that cluster.

### 4.3 Sentence Reordering

It is essential for MDS systems in the extraction framework to choose the order in which sentences

should be presented in the final summary. Presenting more important information earlier in a summary is a general strategy for most domains, although importance may be difficult to determine reliably. Similar to (Barzilay and Lee, 2004), we automatically learn how to order our biographical sentences by observing the typical order of presentation of information in a particular domain. We observe that our Wikipedia biographies tend to follow a general presentation template, in which birth information is mentioned before death information, information about current professional position and affiliations usually appear early in the biography, and nuclear family members are typically mentioned before more distant relations. Learning how to order information from these biographies however would require that we learn to identify particular types of biographical information in sentences.

We directly use the position of each sentence in each Wikipedia biography as a way of determining where sentences containing similar information about different target individuals should appear in their biographies. We represent the absolute position of each sentence in its biography as an integer and train an SVM regression model with RBF kernel, from the class/lexical features of the sentence to its position. We represent each sentence by a feature vector whose elements correspond to the frequency of unigrams and bigrams of class-based items (e.g., GPE, PER) (cf. Section 3) and lexical items; for example, the unigrams *born*, *became*, and *[GPE\_State-or-Province]*, and the bigrams *was born*, *[TARGET\_PER] died* and *[TARGET\_PER] joined* would be good candidates for such features.

To minimize the dimensionality of our regression space, we constrained our feature choice to those features that are important to distinguish biographical sentences, which we term **biographical terms**. Since we want these biographical terms to impact the regression function, we define these to be phrases that consist of at least one lexical item that occurs in many biographies but rarely more than once in any given biography. We compute the biographical term score as in the following equation:

$$bio\_score(t) = \frac{|D_t|}{|D|} \cdot \frac{\sum_{d \in D_t} (1 - \frac{n(t)_d}{\max_t(n(t)_d)})}{|D|} \quad (1)$$

where  $D$  is the set of 16,906 Wikipedia biographies,

$n(t)_d$  is the number of occurrences of term  $t$  in document  $d$ , and  $D_t = \{d \in D : t \in d\}$ . The left factor represents the document frequency of term  $t$ , and the right factor calculates how infrequent the term is in each biography that contains  $t$  at least once.<sup>2</sup> We order the unigrams and bigrams in the biographies by their biographical term scores and select the highest 1K unigrams and 500 bigrams; these thresholds were determined empirically.

#### 4.4 Reference Rewriting

We observe that news articles typically mention biographical information that occurs early in Wikipedia biographies when they mention individuals for the first time in a story (e.g. *Stephen Hawking, the Cambridge University physicist*). We take advantage of the fact that the coreference resolution system we use tags full noun phrases including appositives as part of NEs. Therefore, we initially search for the sentence that contains the longest identified NE (of type PER) that includes the target person’s full name and is coreferential with the target according to the reference resolution system; we denote this NE *NE-NP*. If this sentence has already been classified as a biographical sentence by our classifier, we simply boost its rank in the summary to first. Otherwise, when we order our sentences, we replace the reference to the target person in the first sentence by *NE-NP*. For example, if the first sentence in the biography we have produced for Jimmy Carter is *He was born in 1947* and a sentence not chosen for inclusion in our biography *Jimmy Carter, former U.S. President, visited the University of California last year* contains the *NE-NP*, and *Jimmy Carter* and *He* are coreferential, then the first sentence in our biography will be rewritten as *Jimmy Carter, former U.S. President, was born in 1947*. Note that, in the evaluations presented in Section 5, sentence order was modified by this process in only eight summaries.

## 5 Evaluation

To evaluate our biography generation system, we use the document sets created for the biography evalua-

<sup>2</sup>We considered various approaches to feature selection here, such as comparing term frequency between our biographical and non-biographical corpora. However, terms such as *killed* and *died*, which are useful biographical terms, also occur frequently in our non-biographical corpus.

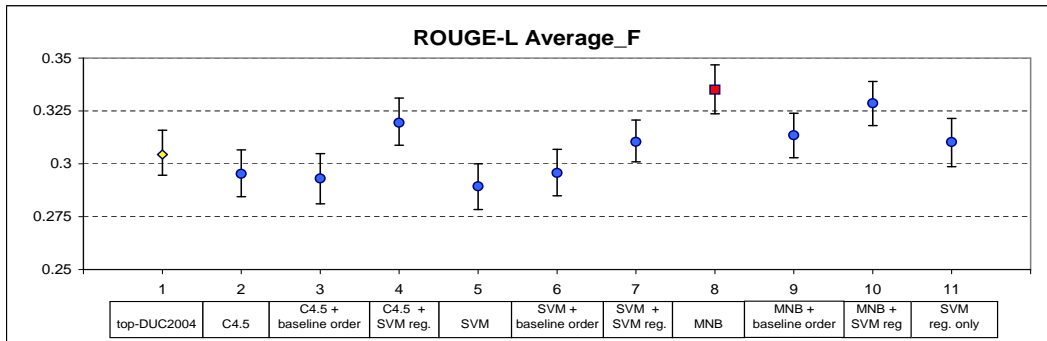


Figure 1: Comparing our approaches against the top performing system in DUC2004 according to ROUGE-L (diamond).

tion (task 5) of DUC2004.<sup>3</sup> The task for systems participating in this evaluation was “*Given each document cluster and a question of the form “Who is X?”, where X is the name of a person or group of people, create a short summary (no longer than 665 bytes) of the cluster that responds to the question.*” NIST assessors chose 50 clusters of TREC documents such that all the documents in a given cluster provide at least part of the answer to this question. Each cluster contained on average 10 documents. NIST had 4 human summaries written for each cluster. A baseline summary was also created for each cluster by extracting the first 665 bytes of the most recent document in the cluster. 22 systems participated in the competition, producing a total of 22 automatic summaries (restricted to 665 bytes) for each cluster. We evaluate our system against the top performing of these 22 systems, according to ROUGE-L, which we denote *top-DUC2004*.<sup>4</sup>

### 5.1 Automatic Evaluation Using ROUGE

As noted in Section 4.1, we experimented with a number of learning algorithms when building our biographical-sentence classifier. For each machine learning algorithm tested, we build a system that initially classifies the input list of sentences into biographical and non-biographical sentences and then

removes redundant sentences. Next, we produce three versions of each system: one which implements a baseline ordering procedure, in which sentences from the clusters are ordered by their appearance in their source document (e.g. any sentence which occurred first in its original document is placed first in the summary, with ties ordered randomly within the set), a second which orders the biographical sentences by the confidence score obtained from the classifier, and a third which uses the SVM regression as the reordering component. Finally, we run our reference rewriting component on each and trim the output to 665 bytes.

We evaluate first using the ROUGE-L metric (Lin and Hovy, 2003) with a 95% (ROUGE computed) confidence interval for all systems and compared these to the ROUGE-L score of the best-performing DUC2004 system.<sup>5</sup> The higher the ROUGE score, the closer the summary is to the DUC2004 human reference summaries. As shown in Figure 1, our best performing system is the multinomial naïve Bayes classifier (MNB) using the classifier confidence scores to order the sentences in the biography. This system significantly outperforms the top ranked DUC2004 system (*top-DUC2004*).<sup>6</sup> The success of this particularly learning algorithm on our task may be due to: (1) the nature of our feature space – n-gram frequencies are modeled properly by a multinomial distribution; (2) the simplicity of this classifier particularly given our large feature dimensional-

<sup>3</sup><http://duc.nist.gov/duc2004>

<sup>4</sup>Note that this system out-performed 19 of the 22 systems on ROUGE-1 and 20 of 22 on ROUGE-L and ROUGE-W-1.2 ( $p < .05$ ) (Blair-Goldensohn et al., 2004a). No ROUGE metric produced scores where this system scored significantly worse than any other system. See Figure 2 below for a comparison of all DUC2004 systems with our top system where all systems are evaluated using ROUGE-L-1.5.5.

<sup>5</sup>We used the same version (1.5.5) of the ROUGE metric to compute scores for the DUC systems and baseline also.

<sup>6</sup>Significance for each pair of systems was determined by paired t-test and calculated at the .05 significance level.

ity; and (3) the robustness of naïve Bayes with respect to noisy data: Not all sentences in Wikipedia biographies are biographical sentences and some sentences in TDT4 *are* biographical.

While the SVM regression reordering component has a slight negative impact on the performance of the MNB system, the difference between the two versions is not significant. Note however, that both the C4.5 and the SVM versions of our system *are* improved by the SVM regression sentence reordering. While neither performs better than top-DUC2004 without this component, the C4.5 system with SVM reordering is significantly better than top-DUC2004 and the performance of the SVM system with SVM regression is comparable to top-DUC2004. In fact, when we use only the SVM regression model to rank the hypothesis sentences, **without** employing any classifier, then remove redundant sentences, rewrite and trim the results, we find that, interestingly, this approach also outperforms top-DUC2004, although the difference is not statistically significant. However, we believe that this is an area worth pursuing in future, with more sophisticated features.

The following biography of Brian Jones was produced by our MNB system and then the sentences were ordered using the SVM regression model:

Born in Bristol in 1947, Brian Jones, the co-pilot on the Breitling mission, learned to fly at 16, dropping out of school a year later to join the Royal Air Force. After earning his commercial balloon flying license, Jones became a ballooning instructor in 1989 and was certified as an examiner for balloon flight licenses by the British Civil Aviation Authority. He helped organize Breitling's most recent around-the-world attempts, in 1997 and 1998. Jones, 52, replaces fellow British flight engineer Tony Brown. Jones, who is to turn 52 next week, is actually the team's third co-pilot. After 13 years of service, he joined a catering business and, in the 1980s,...

Figure 2 illustrates the performance of our MNB system with classifier confidence score sentence ordering when compared to mean ROUGE-L-1.5.5 scores of DUC2004 human-generated summaries and the 22 DUC2004 systems' summaries across all summary tasks. Human summaries are labeled A-H, DUC2004 systems 1-22, and our MNB system is marked by the rectangle. Results are sorted by mean ROUGE-L score. Note that our system performance is actually comparable in ROUGE-L score to one of the human summary generators and is signif-

icantly better than all DUC2004 systems, including top-DUC2004, which is System 1 in the figure.

## 5.2 Manual Evaluation

ROUGE evaluation is based on n-gram overlap between the automatically produced summary and the human reference summaries. Thus, it is not able to measure how fluent or coherent a summary is. Sentence ordering is one factor in determining fluency and coherence. So, we conducted two experiments to measure these qualities, one comparing our top-performing system according to ROUGE-L score (MNB) vs. the top-performing DUC2004 system (top-DUC2004) and another comparing our top system with two different ordering methods, classifier-based and SVM regression.<sup>7</sup> In each experiment, summaries were trimmed to 665 bytes.

In the first experiment, three native American English speakers were presented with the 50 questions (Who is X?). For each question they were given a pair of summaries (presented in random order): one was the output of our MNB system and the other was the summary produced by the top-DUC2004 system. Subjects were asked to decide which summary was more responsive in form and content to the question or whether both were equally responsive. 85.3% (128/150) of subject judgments preferred one summary over the other. 100/128 (78.1%) of these judgments preferred the summaries produced by our MNB system over those produced by top-DUC2004. If we compute the majority vote, there were 42/50 summaries in which at least two subjects made the same choice. 37/42 (88.1%) of these majority judgments preferred our system's summary (using binomial test,  $p = 4.4e - 7$ ). We used the weighted kappa statistic with quadratic weighting (Cohen, 1968) to determine the inter-rater agreement, obtaining a mean pairwise  $\kappa$  of 0.441.

Recall from Section 5.1 that our SVM regression reordering component slightly decreases the average ROUGE score (although not significantly) for our MNB system. For our human evaluations, we decided to evaluate the quality of the presentation of our summaries with and without this compo-

<sup>7</sup>Note that top-DUC2004 was ranked sixth in the DUC 2004 **manual** evaluation, with **no** system performing significantly better for coverage and only 1 system performing significantly better for responsiveness.



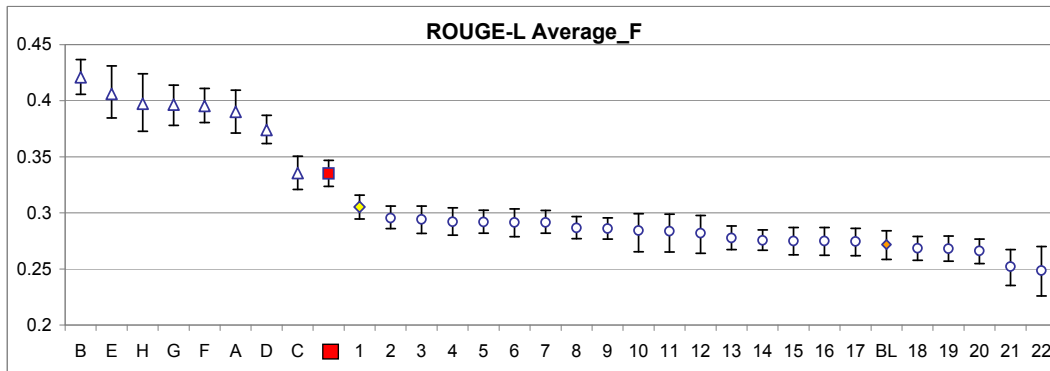


Figure 2: ROUGE-L scores for DUC2004 human summaries (A-H), our MNB system (rectangle), and the DUC2004 competing systems (1-22 anonymized), with the baseline system labeled BL.

ment to see if this reordering component affected human judgments even if it did not improve ROUGE scores. For each question, we produced two summaries from the sentences classified as biographical by the MNB classifier, one ordered by the confidence score obtained by the MNB, in decreasing order, and the other ordered by the SVM regression values, in increasing order. Note that, in three cases, the summary sentences were ordered identically by both procedures, so we used only 47 summaries for this evaluation. Three (different) native American English speakers were presented with the 47 questions for which sentence ordering differed. For each question they were given the two summaries (presented in random order) and asked to determine which biography they preferred.

We found inter-rater agreement for these judgments using Fleiss’ kappa (Fleiss, 1971) to be only moderate ( $\kappa=0.362$ ). However, when we computed the majority vote for each question, we found that 61.7% (29/47) preferred the SVM regression ordering over the MNB classifier confidence score ordering. Although this difference is not statistically significant, again we find the SVM regression ordering results encouraging enough to motivate our further research on improving such ordering procedures.

## 6 Related Work

The DUC2004 system achieving the highest overall ROUGE score, our top-DUC2004 in Section 5, was Blair-Goldensohn et al. (2004a)’s DefScriber, which treats “Who is X?” as a definition question and targets definitional themes (e.g. genus-species)

found in the input document collections which include references to the target person. Extracted sentences are then rewritten using a reference rewriting system (Nenkova and McKeown, 2003) which attempts to shorten subsequent references to the target. Sentences are ordered in the summary based on a weighted combination of topic centrality, lexical cohesion, and topic coverage scores. A similar approach is explored in Biryukov et al. (2005), which uses Topic Signatures (Lin and Hovy, 2000) constructed around the target individual’s name to identify sentences to be included in the biography.

Zhou et al. (2004)’s biography generation system, like ours, trains biographical and non-biographical sentence classifiers to select sentences to be included in the biography. Their system is trained on a hand-annotated corpus of 130 biographies of 12 people, tagged with 9 biographical elements (e.g., bio, education, nationality) and uses binary unigram and bigram lexical and unigram part-of-speech features for classification. Duboue et al. (2003) also address the problem of learning content selection rules for biography. They learn rules from two corpora, a semi-structured corpus with lists of biographical facts about show business celebrities and a corpus of free-text biographies about the same celebrities.

Filatova et al. (2005) learn text features typical of biographical descriptions by deducing biographical and occupation-related activities automatically by comparing descriptions of people with different occupations. Weischedel et al. (2004) models kernel-fact features typical for biographies using linguistic and semantic processing. Linguistic features

are derived from predicate-argument structures deduced from parse trees, and semantic features are the set of biography-related relations and events defined in the ACE guidelines (Dodgington et al., 2004). Sentences containing kernel facts are ranked using probabilities estimated from a corpus of manually created biographies, including Wikipedia, to estimate the conditional distribution of relevant material given a kernel fact and a background corpus.

The problem of ordering sentences and preserving coherence in MDS is addressed by Barzilay et al. (2001), who combine chronological ordering of events with cohesion metrics. SVM regression has recently been used by (Li et al., 2007) for sentence ranking for general MDS. The authors calculated a similarity score for each sentence to the human summaries and then regress numeric features (e.g., the centroid) from each sentence to this score. Barzilay and Lee (2004) use HMMs to capture topic shift within a particular domain; sequence of topic shifts then guides the subsequent ordering of sentences within the summary.

## 7 Discussion and Future Work

In this paper, we describe a MDS system for producing biographies, given a target name. We present an unsupervised approach using Wikipedia biography pages and a general news corpus (TDT4) to automatically construct training data for our system. We employ a NE tagger and a coreference resolution system to extract class-based and lexical features from each sentence which we use to train a binary classifier to identify biographical sentences. We also train an SVM regression model to reorder the sentences and then employ a rewriting heuristic to create the final summary.

We compare versions of our system based upon three machine learning algorithms and two sentence reordering strategies plus a baseline. Our best performing system uses the multinomial naïve Bayes (MNB) classifier with classifier confidence score reordering. However, our SVM regression reordering improves summaries produced by the other two classifiers and is preferred by human judges. We compare our MNB system on the DUC2004 biography task (task 5) to other DUC2004 systems and to human-generated summaries. Our system

out-performs all DUC2004 systems significantly, according to ROUGE-L-1.5.5. When presented with summaries produced by our system and summaries produced by the best-performing (according to ROUGE scores) of the DUC2004 systems, human judges (majority vote of 3) prefer our system’s biographies in 88.1% of cases.

In addition to its high performance, our approach has the following advantages: It employs no manual annotation but relies upon identifying appropriately different corpora to represent our training corpus. It employs class-based as well as lexical features where the classes are obtained automatically from an ACE NE tagger. It utilizes automatic coreference resolution to identify sentences containing references to the target person. Our sentence reordering approaches make use of either classifier confidence scores or ordering learned automatically from the actual ordering of sentences in Wikipedia biographies to determine the order of presentation of sentences in our summaries.

Since our task is to produce concise summaries, one focus of our future research will be to simplify the sentences we extract before classifying them as biographical or non-biographical. This procedure should also help to remove irrelevant information from sentences. Recall that our SVM regression model for sentence ordering was trained using only biographical class-based/lexical items. In future, we would also like to experiment with more linguistically-informed features. While Wikipedia does not enforce any particular ordering of information in biographies, and while different biographies may emphasize different types of information, it would appear that the success of our automatically derived ordering procedures may capture some underlying shared view of how biographies are written. The same underlying views may also apply to domains such as organization descriptions or types of historical events. In future we plan to explore such a generalization of our procedures to such domains.

## Acknowledgments

We thank Kathy McKeown, Andrew Rosenberg, Wisam Dakka, and the Speech and NLP groups at Columbia for useful discussions. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001106C0023 (approved for public release, distribution unlimited). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

## References

- Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of NAACL-HLT*.
- Regina Barzilay, Noemie Elhadad, and Kathleen McKeown. 2001. Sentence ordering in multidocument summarization. In *Proceedings of the First Human Language Technology Conference*, San Diego, California.
- Maria Biryukov, Roxana Angheluta, and Marie-Francine Moens. 2005. Multidocument question answering text summarization using topic signatures. In *Proceedings of the 5th Dutch-Belgium Information Retrieval Workshop*, Utrecht, the Netherlands.
- Sasha Blair-Goldensohn, David Evans, Vasileios Hatzivassiloglou, Kathleen McKeown, Ani Nenkova, Rebecca Passonneau, Barry Schiffman, Andrew Schlaikjer, Advait Siddharthan, and Sergey Siegelman. 2004a. Columbia University at DUC 2004. In *Proceedings of the 4th Document Understanding Conference*, Boston, Massachusetts, USA.
- Sasha Blair-Goldensohn, Kathy McKeown, and Andrew Schlaikjer. 2004b. Answering definitional questions: A hybrid approach. In Mark Maybury, editor, *New Directions In Question Answering*, chapter 4. AAAI Press.
- J. Cohen. 1968. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. volume 70, pages 213–220.
- George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction program - tasks, data, and evaluation. In *Proceedings of the LREC Conference*, Canary Islands, Spain, July.
- Pablo Duboue and Kathleen McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing*, pages 121–128, Sapporo, Japan, July.
- Elena Filatova and John Prager. 2005. Tell me what you do and I'll tell you what you are: Learning occupation-related activities for biographies. In *Proceedings of the Joint Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 113–120, Vancouver, Canada, October.
- J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. volume 76, No. 5, pages 378–382.
- Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyu's english ace 2005 system description. In *ACE 05 Evaluation Workshop*, Gaithersburg, MD.
- Sujian Li, You Ouyang, Wei Wang, and Bin Sun. 2007. Multi-document summarization using support vector regression. In <http://duc.nist.gov/pubs/2007papers>.
- Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 495–501, Saarbrücken, Germany, July.
- Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Language Technology Conference*, Edmonton, Canada.
- Ani Nenkova and Kathleen McKeown. 2003. References to named entities: A corpus study. In *Proceedings of the Joint Human Language Technology Conference and North American chapter of the Association for Computational Linguistics Annual Meeting*, Edmonton, Canada, May.
- Ralph Weischedel, Jinxi Xu, and Ana Licuanan. 2004. A hybrid approach to answering biographical questions. In Mark Maybury, editor, *New Directions In Question Answering*, chapter 5. AAAI Press.
- I. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. Cunningham. 1999. Weka: Practical machine learning tools and techniques with java implementation. In *International Workshop: Emerging Knowledge Engineering and Connectionist-Based Information Systems*, pages 192–196.
- Liang Zhou, Miruna Ticea, and Eduard Hovy. 2004. Multi-document biography summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 434–441, Barcelona, Spain.

# Generating Impact-Based Summaries for Scientific Literature

**Qiaozhu Mei**

University of Illinois at Urbana-  
Champaign  
qmei2@uiuc.edu

**ChengXiang Zhai**

University of Illinois at Urbana-  
Champaign  
czhai@cs.uiuc.edu

## Abstract

In this paper, we present a study of a novel summarization problem, i.e., summarizing the impact of a scientific publication. Given a paper and its citation context, we study how to extract sentences that can represent the most influential content of the paper. We propose language modeling methods for solving this problem, and study how to incorporate features such as authority and proximity to accurately estimate the impact language model. Experiment results on a SIGIR publication collection show that the proposed methods are effective for generating impact-based summaries.

## 1 Introduction

The volume of scientific literature has been growing rapidly. From recent statistics, each year 400,000 new citations are added to MEDLINE, the major biomedical literature database<sup>1</sup>. This fast growth of literature makes it difficult for researchers, especially beginning researchers, to keep track of the research trends and find high impact papers on unfamiliar topics.

Impact factors (Kaplan and Nelson, 2000) are useful, but they are just numerical values, so they cannot tell researchers which aspects of a paper are influential. On the other hand, a regular content-based summary (e.g., the abstract or conclusion section of a paper or an automatically generated topical summary (Giles et al., 1998)) can help a user know

about the main content of a paper, but not necessarily the most influential content of the paper. Indeed, the abstract of a paper mostly reflects the expected impact of the paper as perceived by the author(s), which could significantly deviate from the actual impact of the paper in the research community. Moreover, the impact of a paper changes over time due to the evolution and progress of research in a field. For example, an algorithm published a decade ago may be no longer the state of the art, but the problem definition in the same paper can be still well accepted.

Although much work has been done on text summarization (See Section 6 for a detailed survey), to the best of our knowledge, the problem of impact summarization has not been studied before. In this paper, we study this novel summarization problem and propose language modeling-based approaches to solving the problem. By definition, the impact of a paper has to be judged based on the consent of research community, especially by people who cited it. Thus in order to generate an impact-based summary, we must use not only the original content, but also the descriptions of that paper provided in papers which cited it, making it a challenging task and different from a regular summarization setup such as news summarization. Indeed, unlike a regular summarization system which identifies and interprets the *topic* of a document, an impact summarization system should identify and interpret the *impact* of a paper.

We define the impact summarization problem in the framework of extraction-based text summarization (Luhn, 1958; McKeown and Radev, 1995), and cast the problem as an impact sentence retrieval

<sup>1</sup><http://www.nlm.nih.gov/bsd/history/tsld024.htm>

problem. We propose language models to exploit both the citation context and original content of a paper to generate an impact-based summary. We study how to incorporate features such as authority and proximity into the estimation of language models. We propose and evaluate several different strategies for estimating the impact language model, which is key to impact summarization. No existing test collection is available for evaluating impact summarization. We construct a test collection using 28 years of ACM SIGIR papers (1978 - 2005) to evaluate the proposed methods. Experiment results on this collection show that the proposed approaches are effective for generating impact-based summaries. The results also show that using both the original document content and the citation contexts is important and incorporating citation authority and proximity is beneficial.

An impact-based summary is not only useful for facilitating the exploration of literature, but also helpful for suggesting query terms for literature retrieval, understanding the evolution of research trends, and identifying the interactions of different research fields. The proposed methods are also applicable to summarizing the impact of documents in other domains where citation context exists, such as emails and weblogs.

The rest of the paper is organized as follows. In Section 2 and 3, we define the impact-based summarization problem and propose the general language modeling approach. In Section 4, we present different strategies and features for estimating an impact language model, a key challenge in impact summarization. We discuss our experiments and results in Section 5. Finally, the related work and conclusions are discussed in Section 6 and Section 7.

## 2 Impact Summarization

Following the existing work on topical summarization of scientific literature (Paice, 1981; Paice and Jones, 1993), we define an impact-based summary of a paper as a set of sentences extracted from a paper that can reflect the impact of the paper, where “impact” is roughly defined as the influence of the paper on research of similar or related topics as reflected in the citations of the paper. Such an extraction-based definition of summarization has

also been quite common in most existing general summarization work (Radev et al., 2002).

By definition, in order to generate an impact summary of a paper, we must look at how other papers cite the paper, use this information to infer the impact of the paper, and select sentences from the original paper that can reflect the inferred impact. Note that we do not directly use the sentences from the citation context to form a summary. This is because in citations, the discussion of the paper cited is usually mixed with the content of the paper citing it, and sometimes also with discussion about other papers cited (Siddharthan and Teufel, 2007).

Formally, let  $d = (s_0, s_1, \dots, s_n)$  be a paper to be summarized, where  $s_i$  is a sentence. We refer to a sentence (in another paper) in which there is an explicit citation of  $d$  as a *citing sentence* of  $d$ . When a paper is cited, it is often discussed consecutively in more than one sentence near the citation, thus intuitively we would like to consider a window of sentences centered at a citing sentence; the window size would be a parameter to set. We call such a window of sentences a *citation context*, and use  $C$  to denote the union of all the citation contexts of  $d$  in a collection of research papers. Thus  $C$  itself is a set (more precisely bag) of sentences. The task of **impact-based summarization** is thus to 1) construct a representation of the impact of  $d$ ,  $I$ , based on  $d$  and  $C$ ; 2) design a scoring function  $Score(.)$  to rank sentences in  $d$  based on how well a sentence reflects  $I$ . A user-defined number of top-ranked sentences can then be selected as the impact summary for  $d$ .

The formulation above immediately suggests that we can cast the impact summarization problem as a retrieval problem where each candidate sentence in  $d$  is regarded as a “document,” the impact of the paper (i.e.,  $I$ ) as a “query,” and our goal is to “retrieve” sentences that can reflect the impact of the paper as indicated by the citation context. Looking at the problem in this way, we see that there are two main challenges in impact summarization: first, we must be able to *infer* the impact based on both the citation contexts and the original document; second, we should measure how well a sentence reflects this inferred impact. To solve these challenges, in the next section, we propose to model impact with unigram language models and score sentences using

Kullback-Leibler divergence. We further propose methods for estimating the impact language model based on several features including the authority of citations, and the citation proximity.

### 3 Language Models for Impact Summarization

#### 3.1 Impact language models

From the retrieval perspective, our collection is the paper to be summarized, and each sentence is a “document” to be retrieved. However, unlike in the case of ad hoc retrieval, we do not really have a query describing the impact of the paper; instead, we have a lot of citation contexts that can be used to infer information about the query. Thus the main challenge in impact summarization is to effectively construct a “virtual impact query” based on the citation contexts.

What should such a virtual impact query look like? Intuitively, it should model the impact-reflecting content of the paper. We thus propose to represent such a virtual impact query with a unigram language model. Such a model is expected to assign high probabilities to those words that can describe the impact of paper  $d$ , just as we expect a query language model in ad hoc retrieval to assign high probabilities to words that tend to occur in relevant documents (Ponte and Croft, 1998). We call such a language model the *impact language model* of paper  $d$  (denoted as  $\theta_I$ ); it can be estimated based on both  $d$  and its citation context  $C$  as will be discussed in Section 4.

#### 3.2 KL-divergence scoring

With the impact language model in place, we can then adopt many existing probabilistic retrieval models such as the classical probabilistic retrieval models (Robertson and Sparck Jones, 1976) and the Kullback-Leibler (KL) divergence retrieval model (Lafferty and Zhai, 2001; Zhai and Lafferty, 2001a), to solve the problem of impact summarization by scoring sentences based on the estimated impact language model. In our study, we choose to use the KL-divergence scoring method to score sentences as this method has performed well for regular ad hoc retrieval tasks (Zhai and Lafferty, 2001a) and has an information theoretic interpretation.

To apply the KL-divergence scoring method, we assume that a candidate sentence  $s$  is generated from a sentence language model  $\theta_s$ . Given  $s$  in  $d$  and the citation context  $C$ , we would first estimate  $\theta_s$  based on  $s$  and estimate  $\theta_I$  based on  $C$ , and then score  $s$  with the negative KL divergence of  $\theta_s$  and  $\theta_I$ . That is,

$$\begin{aligned} \text{Score}(s) &= -D(\theta_I|\theta_s) \\ &= \sum_{w \in V} p(w|\theta_I) \log p(w|\theta_s) - \sum_{w \in V} p(w|\theta_I) \log p(w|\theta_I) \end{aligned}$$

where  $V$  is the set of words in our vocabulary and  $w$  denotes a word.

From the information theoretic perspective, the KL-divergence of  $\theta_s$  and  $\theta_I$  can be interpreted as measuring the average number of bits wasted in compressing messages generated according to  $\theta_I$  (i.e., impact descriptions) with coding non-optimally designed based on  $\theta_s$ . If  $\theta_s$  and  $\theta_I$  are very close, the KL-divergence would be small and  $\text{Score}(s)$  would be high, which intuitively makes sense. Note that the second term (entropy of  $\theta_I$ ) is independent of  $s$ , so it can be ignored for ranking  $s$ .

We see that according to the KL-divergence scoring method, our main tasks are to estimate  $\theta_s$  and  $\theta_I$ . Since  $s$  can be regarded as a short document, we can use any standard method to estimate  $\theta_s$ . In this work, we use Dirichlet prior smoothing (Zhai and Lafferty, 2001b) to estimate  $\theta_s$  as follows:

$$p(w|\theta_s) = \frac{c(w, s) + \mu_s * P(w|D)}{|s| + \mu_s} \quad (1)$$

where  $|s|$  is the length of  $s$ ,  $c(w, s)$  is the count of word  $w$  in  $s$ ,  $p(w|D)$  is a background model estimated using  $\frac{c(w, D)}{\sum_{w' \in V} c(w', D)}$  ( $D$  can be the set of all the papers available to us) and  $\mu_s$  is a smoothing parameter to be empirically set. Note that as the length of a sentence is very short, smoothing is critical for addressing the data sparseness problem.

The remaining challenge is to estimate  $\theta_I$  accurately based on  $d$  and its citation contexts.

### 4 Estimation of Impact Language Models

Intuitively, the impact of a paper is mostly reflected in the citation context. Thus the estimation of the impact language model should be primarily based on the citation context  $C$ . However, we would like

our impact model to be able to help us select impact-reflecting sentences from  $d$ , thus it is important for the impact model to explain well the paper content in general. To achieve this balance, we treat the citation context  $C$  as prior information and the current document  $d$  as the observed data, and use Bayesian estimation to estimate the impact language model.

Specifically, let  $p(w|C)$  be a citation context language model estimated based on the citation context  $C$ . We define Dirichlet prior with parameters  $\{\mu_C p(w|C)\}_{w \in V}$  for the impact model, where  $\mu_C$  encodes our confidence on this prior and effectively serves as a weighting parameter for balancing the contribution of  $C$  and  $d$  for estimating the impact model. Given the observed document  $d$ , the posterior mean estimate of the impact model would be (MacKay and Peto, 1995; Zhai and Lafferty, 2001b)

$$P(w|\theta_I) = \frac{c(w, d) + \mu_C p(w|C)}{|d| + \mu_C} \quad (2)$$

$\mu_C$  can be interpreted as the equivalent sample size of our prior. Thus setting  $\mu_C = |d|$  means that we put equal weights on the citation context and the document itself.  $\mu_C = 0$  yields  $p(w|\theta_I) = p(w|d)$ , which is to say that the impact is entirely captured by the paper itself, and our impact summarization problem would then become the standard single document (topical) summarization. Intuitively though, we would want to set  $\mu_C$  to a relatively large number to exploit the citation context in our estimation, which is confirmed in our experiments.

An alternative way is to simply interpolate  $p(w|d)$  and  $p(w|C)$  with a constant coefficient:

$$p(w|\theta_I) = (1 - \delta)p(w|d) + \delta p(w|C) \quad (3)$$

We will compare the two strategies in Section 5.

How do we estimate  $p(w|C)$ ? Intuitively, words occurring in  $C$  frequently should have high probabilities. A simple way is to pool together all the sentences in  $C$  and use the maximum likelihood estimator,

$$p(w|C) = \frac{\sum_{s \in C} c(w, s)}{\sum_{w' \in V} \sum_{s' \in C} c(w', s')} \quad (4)$$

where  $c(w, s)$  is the count of  $w$  in  $s$ .

One deficiency of this simple estimate is that we treat all the (extended) citation sentences equally.

However, there are at least two reasons why we want to assign unequal weights to different citation sentences: (1) A sentence closer to the citation label should contribute more than one far away. (2) A sentence occurring in a highly authoritative paper should contribute more than that in a less authoritative paper. To capture these two heuristics, we define a weight coefficient  $\alpha_s$  for a sentence  $s$  in  $C$  as follows:

$$\alpha_s = pg(s)pr(s)$$

where  $pg(s)$  is an authority score of the paper containing  $s$  and  $pr(s)$  is a proximity score that rewards a sentence close to the citation label.

For example,  $pg(s)$  can be the PageRank value (Brin and Page, 1998) of the document with  $s$ , which measures the authority of the document based on a citation graph, and is computed as follows: We construct a directed graph from the collection of scientific literature with each paper as a vertex and each citation as a directed edge pointing from the citing paper to the cited paper. We can then use the standard PageRank algorithm (Brin and Page, 1998) to compute a PageRank value for each document. We used this approach in our experiments.

We define  $pr(s)$  as  $pr(s) = \frac{1}{\alpha^k}$ , where  $k$  is the distance (counted in terms of the number of sentences) between sentence  $s$  and the center sentence of the window containing  $s$ ; by ‘‘center sentence’’, we mean the citing sentence containing the citation label. Thus the sentence with the citation label will have a proximity of 1 (because  $k = 0$ ), while the sentences away from the citation label will have a decaying weight controlled by parameter  $\alpha$ .

With  $\alpha_s$ , we can then use the following ‘‘weighted’’ maximum likelihood estimate for the impact language model:

$$p(w|C) = \frac{\sum_{s \in C} \alpha_s c(w, s)}{\sum_{w' \in V} \sum_{s' \in C} \alpha_{s'} c(w', s')} \quad (5)$$

As we will show in Section 5, this weighted maximum likelihood estimate performs better than the simple maximum likelihood estimate, and both  $pg(s)$  and  $pr(s)$  are useful.

## 5 Experiments and Results

### 5.1 Experiment Design

#### 5.1.1 Test set construction

Because no existing test set is available for evaluating impact summarization, we opt to create a test set based on 28 years of ACM SIGIR papers (1978 - 2005) available through the ACM Digital Library<sup>2</sup> and the SIGIR membership. Leveraging the explicit citation information provided by ACM Digital Library, for each of the 1303 papers, we recorded all other papers that cited the paper and extracted the citation context from these citing papers. Each citation context contains 5 sentences with 2 sentences before and after the citing sentence.

Since a low-impact paper would not be useful for evaluating impact summarization, we took all the 14 papers from the SIGIR collection that have no less than 20 citations by papers in the same collection as candidate papers for evaluation. An expert in Information Retrieval field read each paper and its citation context, and manually created an impact-based summary by selecting all the “impact-capturing” sentences from the paper. Specifically, the expert first attempted to understand the most influential content of a paper by reading the citation contexts. The expert then read each sentence of the paper and made a decision whether the sentence covers some “influential content” as indicated in the citation contexts. The sentences that were decided as covering some influential content were then collected as the gold standard impact summary for the paper.

We assume that the title of a paper will always be included in the summary, so we excluded the title both when constructing the gold standard and when generating a summary. The gold standard summaries have a minimum length of 5 sentences and a maximum length of 18 sentences; the median length is 9 sentences. These 14 impact-based summaries are used as gold standards for our experiments, based on which all summaries generated by the system are evaluated. This data set is available at <http://timan.cs.uiuc.edu/data/impact.html>. We must admit that using only 14 papers and only one expert for evaluation is a limitation of our work. However,

<sup>2</sup><http://www.acm.org/dl>

going beyond the 14 papers would risk reducing the reliability of impact judgment due to the sparseness of citations. How to develop a better test collection is an important future direction.

#### 5.1.2 Evaluation Metrics

Following the current practice in evaluating summarization, particularly DUC<sup>3</sup>, we use the ROUGE evaluation package (Lin and Hovy, 2003). Among ROUGE metrics, ROUGE-N (models n-gram co-occurrence,  $N = 1, 2$ ) and ROUGE-L (models longest common sequence) generally perform well in evaluating both single-document summarization and multi-document summarization (Lin and Hovy, 2003). Since they are general evaluation measures for summarization, they are also applicable to evaluating the MEAD-Doc+Cite baseline method to be described below. Thus although we evaluated our methods with all the metrics provided by ROUGE, we only report ROUGE-1 and ROUGE-L in this paper (other metrics give very similar results).

#### 5.1.3 Baseline methods

Since impact summarization has not been previously studied, there is no natural baseline method to compare with. We thus adapt some state-of-the-art conventional summarization methods implemented in the MEAD toolkit (Radev et al., 2003)<sup>4</sup> to obtain three baseline methods: (1) **LEAD**: It simply extracts sentences from the beginning of a paper, i.e., sentences in the abstract or beginning of the introduction section; we include **LEAD** to see if such “leading sentences” reflect the impact of a paper as authors presumably would expect to summarize a paper’s contributions in the abstract. (2) **MEAD-Doc**: It uses the single-document summarizer in MEAD to generate a summary based solely on the original paper; comparison with this baseline can tell us how much better we can do than a conventional topic-based summarizer that does not consider the citation context. (3) **MEAD-Doc+Cite**: Here we concatenate all the citation contexts in a paper to form a “citation document” and then use the MEAD multidocument summarizer to generate a summary from the original paper plus all its citation documents; this baseline represents a reasonable way

<sup>3</sup><http://duc.nist.gov/>

<sup>4</sup><http://www.summarization.com/mead/>



Sum. Length	Metric	Random	LEAD	MEAD-Doc	MEAD-Doc+Cite	KL-Divergence
3	ROUGE-1	0.163	0.167	0.301*	0.248	<b>0.323</b>
3	ROUGE-L	0.144	0.158	0.265	0.217	<b>0.299</b>
5	ROUGE-1	0.230	0.301	0.401	0.333	<b>0.467</b>
5	ROUGE-L	0.214	0.292	0.362	0.298	<b>0.444</b>
10	ROUGE-1	0.430	0.514	0.575	0.472	<b>0.649</b>
10	ROUGE-L	0.396	0.494	0.535	0.428	<b>0.622</b>
15	ROUGE-1	0.538	0.610	0.685	0.552	<b>0.730</b>
15	ROUGE-L	0.499	0.586	0.650	0.503	<b>0.705</b>

Table 1: Performance Comparison of Summarizers

of applying an existing summarization method to generate an impact-based summary. Note that this method may extract sentences in the citation contexts but not in the original paper.

## 5.2 Basic Results

We first show some basic results of impact summarization in Table 1. They are generated using constant coefficient interpolation for the impact language model (i.e., Equation 3) with  $\delta = 0.8$ , weighted maximum likelihood estimate for the citation context model (i.e., Equation 5) with  $\alpha = 3$ , and  $\mu_s = 1,000$  for candidate sentence smoothing (Equation 1). These results are not necessarily optimal as will be seen when we examine parameter and method variations.

From Table 1, we see clearly that our method consistently outperforms all the baselines. Among the baselines, MEAD-Doc is consistently better than both LEAD and MEAD-Doc+Cite. While MEAD-Doc’s outperforming LEAD is not surprising, it is a bit surprising that MEAD-Doc also outperforms MEAD-Doc+Cite as the latter uses both the citation context and the original document. One possible explanation may be that MEAD is not designed for impact summarization and it has been trapped by the distracting content in the citation context<sup>5</sup>. Indeed, this can also explain why MEAD-Doc+Cite tends to perform worse than LEAD by ROUGE-L since if MEAD-Doc+Cite picks up sentences from the citation context rather than the original papers, it would not match as well with the gold standard as LEAD which selects sentences from the origi-

<sup>5</sup>One anonymous reviewer suggested an interesting improvement to the MEAD-Doc+Cite baseline, in which we would first extract sentences from the citation context and then for each extracted sentence find a similar one in the original paper. Unfortunately, we did not have time to test this approach before the deadline for the camera-ready version of this paper.

nal papers. These results thus show that conventional summarization techniques are inadequate for impact summarization, and the proposed language modeling methods are more effective for generating impact-based summaries.

In Table 2, we show a sample impact-based summary and the corresponding MEAD-Doc regular summary. We see that the regular summary tends to have general sentences about the problem, background and techniques, not very informative in conveying specific contributions of the paper. None of these sentences was selected by the human expert. In contrast, the sentences in the impact summary cover several details of the impact of the paper (i.e., specific smoothing methods especially Dirichlet prior, sensitivity of performance to smoothing, and dual role of smoothing), and sentences 4 and 6 are also among the 8 sentences picked by the human expert. Interestingly, neither sentence is in the abstract of the original paper, suggesting a deviation of the actual impact of a paper and that perceived by the author(s).

## 5.3 Component analysis

We now turn to examine the effectiveness of each component in the proposed methods and different strategies for estimating  $\theta_I$ .

**Effectiveness of interpolation:** We hypothesized that we need to use both the original document and the citation context to estimate  $\theta_I$ . To test this hypothesis, we compare the results of using only  $d$ , only the citation context, and interpolation of them in Table 3. We show two different strategies of interpolation (i.e., constant coefficient with  $\delta = 0.8$  and Dirichlet with  $\mu_c = 20,000$ ) as described in Section 4.

From Table 3, we see that both strategies of interpolation indeed outperform using either the origi-

Impact-based summary:
1. Figure 5: Interpolation versus backoff for Jelinek-Mercer (top), Dirichlet smoothing (middle), and absolute discounting (bottom).
2. Second, one can de-couple the two different roles of smoothing by adopting a two stage smoothing strategy in which Dirichlet smoothing is first applied to implement the estimation role and Jelinek-Mercer smoothing is then applied to implement the role of query modeling
3. We find that the backoff performance is more sensitive to the smoothing parameter than that of interpolation, especially in Jelinek-Mercer and Dirichlet prior.
4. We then examined three popular interpolation-based smoothing methods (Jelinek-Mercer method, Dirichlet priors, and absolute discounting), as well as their backoff versions, and evaluated them using several large and small TREC retrieval testing collections.
summary 5. By rewriting the query-likelihood retrieval model using a smoothed document language model, we derived a general retrieval formula where the smoothing of the document language model can be interpreted in terms of several heuristics used intraditional models, including TF-IDF weighting and document length normalization.
6. We find that the retrieval performance is generally sensitive to the smoothing parameters, suggesting that an understanding and appropriate setting of smoothing parameters is very important in the language modeling approach.
Regular summary (generated using MEAD-Doc):
1. Language modeling approaches to information retrieval are attractive and promising because they connect the problem of retrieval with that of language model estimation, which has been studied extensively in other application areas such as speech recognition.
2. The basic idea of these approaches is to estimate a language model for each document, and then rank documents by the likelihood of the query according to the estimated language model.
3. On the one hand, theoretical studies of an underlying model have been developed; this direction is, for example, represented by the various kinds of logic models and probabilistic models (e.g., [14, 3, 15, 22]).
4. After applying the Bayes' formula and dropping a document-independent constant (since we are only interested in ranking documents), we have $p(d q) \propto (q d)p(d)$ .
5. As discussed in [1], the righthand side of the above equation has an interesting interpretation, where, $p(d)$ is our prior belief that $d$ is relevant to any query and $p(q d)$ is the query likelihood given the document, which captures how well the document "fits" the particular query $q$ .
6. The probability of an unseen word is typically taken as being proportional to the general frequency of the word, e.g., as computed using the document collection.

Table 2: Impact-based summary vs. regular summary for the paper "A study of smoothing methods for language models applied to ad hoc information retrieval".

nal document model ( $p(w|d)$ ) or the citation context model ( $p(w|C)$ ) alone, which confirms that both the original paper and the citation context are important for estimating  $\theta_I$ . We also see that using the citation context alone is better than using the original paper alone, which is expected. Between the two strategies, Dirichlet dynamic coefficient is slightly better than constant coefficient (CC), after optimizing the interpolation parameter for both strategy.

Measure	$P(w d)$	$P(w C)$	Interpolation	
			ConstCoef	Dirichlet
ROUGE-1	0.529	0.635	0.643	<b>0.647</b>
ROUGE-L	0.501	0.607	0.619	<b>0.623</b>

Table 3: Effectiveness of interpolation

**Citation authority and proximity:** These heuristics are very interesting to study as they are unique to impact summarization and not well studied in the existing summarization work.

pg(s)	pr(s) off	pr(s)=1/ $\alpha^k$		
		$\alpha = 2$	$\alpha = 3$	$\alpha = 4$
<b>Off</b>	0.685	0.711	0.714	0.700
<b>On</b>	0.708	0.712	0.706	0.703

Table 4: Authority (pg(s)) and proximity (pr(s))

In Table 4, we show the ROUGE-L values for various combinations of these two heuristics (summary length is 15). We turn off either  $pg(s)$  or  $pr(s)$  by setting it to a constant; when both are turned off, we have the unweighted MLE of  $p(w|C)$  (Equation 4). Clearly, using weighted MLE with any of the two heuristics is better than the unweighted MLE, indicating that both heuristics are effective. However, combining the two heuristics does not always improve over using a single one. Since intuitively these two heuristics are orthogonal, this may suggest that our way of combining the two scores (i.e., taking a product of them) may not be optimal; further study is needed to better understand this. The ROUGE-1 results are similar.

**Tuning of other parameters:** There are three other parameters which need to be tuned: (1)  $\mu_s$  for candidate sentence smoothing (Equation 1); (2)  $\mu_c$  in Dirichlet interpolation for impact model estimation (Equation 2); and (3)  $\delta$  in constant coefficient interpolation (Equation 3). We have examined the sensitivity of performance to these parameters. In general, for a wide range of values of these parameters, the performance is relatively stable and near optimal. Specifically, the performance is near optimal as

long as  $\mu_s$  and  $\mu_c$  are sufficiently large ( $\mu_s \geq 1000$ ,  $\mu_c \geq 20,000$ ), and the interpolation parameter  $\delta$  is between 0.4 and 0.9.

## 6 Related Work

General text summarization, including single document summarization (Luhn, 1958; Goldstein et al., 1999) and multi-document summarization (Kraaij et al., 2001; Radev et al., 2003) has been well studied; our work is under the framework of extractive summarization (Luhn, 1958; McKeown and Radev, 1995; Goldstein et al., 1999; Kraaij et al., 2001), but our problem formulation differs from any existing formulation of the summarization problem. It differs from regular single-document summarization because we utilize extra information (i.e. citation contexts) to summarize the impact of a paper. It also differs from regular multi-document summarization because the roles of original documents and citation contexts are not equivalent. Specifically, citation contexts serve as an indicator of the impact of the paper, but the summary is generated by extracting the sentences from the original paper.

Technical paper summarization has also been studied (Paice, 1981; Paice and Jones, 1993; Saggion and Lapalme, 2002; Teufel and Moens, 2002), but the previous work did not explore citation context to emphasize the impact of papers.

Citation context has been explored in several studies (Nakov et al., 2004; Ritchie et al., 2006; Schwartz et al., 2007; Siddharthan and Teufel, 2007). However, none of the previous studies has used citation context in the same way as we did, though the potential of *directly* using citation sentences (called *citances*) to summarize a paper was pointed out in (Nakov et al., 2004).

Recently, people have explored various types of auxiliary knowledge such as hyperlinks (Delort et al., 2003) and clickthrough data (Sun et al., 2005), to summarize a webpage; such work is related to ours as anchor text is similar to citation context, but it is based on a standard formulation of multi-document summarization and would contain only sentences from anchor text.

Our work is also related to work on using language models for retrieval (Ponte and Croft, 1998; Zhai and Lafferty, 2001b; Lafferty and Zhai, 2001)

and summarization (Kraaij et al., 2001). However, we do not have an explicit query and constructing the impact model is a novel exploration. We also proposed new language models to capture the impact.

## 7 Conclusions

We have defined and studied the novel problem of summarizing the impact of a research paper. We cast the problem as an impact sentence retrieval problem, and proposed new language models to model the impact of a paper based on both the original content of the paper and its citation contexts in a literature collection with consideration of citation authority and proximity.

To evaluate impact summarization, we created a test set based on ACM SIGIR papers. Experiment results on this test set show that the proposed impact summarization methods are effective and outperform several baselines that represent the existing summarization methods.

An important future work is to construct larger test sets (e.g., of biomedical literature) to facilitate evaluation of impact summarization. Our formulation of the impact summarization problem can be further improved by going beyond sentence retrieval and considering factors such as redundancy and coherency to better organize an impact summary. Finally, automatically generating impact-based summaries can not only help users access and digest influential research publications, but also facilitate other literature mining tasks such as milestone mining and research trend monitoring. It would be interesting to explore all these applications.

## Acknowledgments

We are grateful to the anonymous reviewers for their constructive comments. This work is in part supported by a Yahoo! Graduate Fellowship and NSF grants under award numbers 0713571, 0347933, and 0428472.

## References

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.

- J.-Y. Delort, B. Bouchon-Meunier, and M. Rifqi. 2003. Enhanced web document summarization using hyperlinks. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, pages 208–215.
- C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 1998. Citeseer: an automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, pages 89–98.
- Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. 1999. Summarizing text documents: sentence selection and evaluation metrics. In *Proceedings of ACM SIGIR 99*, pages 121–128.
- Nancy R. Kaplan and Michael L. Nelson. 2000. Determining the publication impact of a digital library. *J. Am. Soc. Inf. Sci.*, 51(4):324–339.
- W. Kraaij, M. Spitters, and M. van der Heijden. 2001. Combining a mixture language model and naive bayes for multi-document summarisation. In *Proceedings of the DUC2001 workshop*.
- John Lafferty and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of ACM SIGIR 2001*, pages 111–119.
- Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 71–78.
- H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.
- D. MacKay and L. Peto. 1995. A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):289–307.
- Kathleen McKeown and Dragomir R. Radev. 1995. Generating summaries of multiple news articles. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–82.
- P. Nakov, A. Schwartz, and M. Hearst. 2004. Citances: Citation sentences for semantic analysis of bioscience text. In *Proceedings of ACM SIGIR'04 Workshop on Search and Discovery in Bioinformatics*.
- Chris D. Paice and Paul A. Jones. 1993. The identification of important concepts in highly structured technical papers. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 69–78.
- C. D. Paice. 1981. The automatic generation of literature abstracts: an approach based on the identification of self-indicating phrases. In *Proceedings of the 3rd Annual ACM Conference on Research and Development in Information Retrieval*, pages 172–191.
- Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281.
- Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. *Comput. Linguist.*, 28(4):399–408.
- Dragomir R. Radev, Simone Teufel, Horacio Saggion, Wai Lam, John Blitzer, Hong Qi, Arda Celebi, Danyu Liu, and Elliott Drabek. 2003. Evaluation challenges in large-scale document summarization: the mead project. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 375–382.
- A. Ritchie, S. Teufel, and S. Robertson. 2006. Creating a test collection for citation-based ir experiments. In *Proceedings of the HLT-NAACL 2006*, pages 391–398.
- S. Robertson and K. Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146.
- Hpracop Saggion and Guy Lapalme. 2002. Generating indicative-informative summaries with sumUM. *Computational Linguistics*, 28(4):497–526.
- A. S. Schwartz, A. Divoli, and M. A. Hearst. 2007. Multiple alignment of citation sentences with conditional random fields and posterior decoding. In *Proceedings of the 2007 EMNLP-CoNLL*, pages 847–857.
- A. Siddharthan and S. Teufel. 2007. Whose idea was this, and why does it matter? attributing scientific work to citations. In *Proceedings of NAACL/HLT-07*, pages 316–323.
- Jian-Tao Sun, Dou Shen, Hua-Jun Zeng, Qiang Yang, Yuchang Lu, and Zheng Chen. 2005. Web-page summarization using clickthrough data. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 194–201.
- Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Comput. Linguist.*, 28(4):409–445.
- ChengXiang Zhai and John Lafferty. 2001a. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410.
- Chengxiang Zhai and John Lafferty. 2001b. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342.

# Can you summarize this? Identifying correlates of input difficulty for generic multi-document summarization

**Ani Nenkova**

University of Pennsylvania  
Philadelphia, PA 19104, USA  
nenkova@seas.upenn.edu

**Annie Louis**

University of Pennsylvania  
Philadelphia, PA 19104, USA  
lannie@seas.upenn.edu

## Abstract

Different summarization requirements could make the writing of a good summary more difficult, or easier. Summary length and the characteristics of the input are such constraints influencing the quality of a potential summary. In this paper we report the results of a quantitative analysis on data from large-scale evaluations of multi-document summarization, empirically confirming this hypothesis. We further show that features measuring the cohesiveness of the input are highly correlated with eventual summary quality and that it is possible to use these as features to predict the difficulty of new, unseen, summarization inputs.

## 1 Introduction

In certain situations even the best automatic summarizers or professional writers can find it hard to write a good summary of a set of articles. If there is no clear topic shared across the input articles, or if they follow the development of the same event in time for a longer period, it could become difficult to decide what information is most representative and should be conveyed in a summary. Similarly, length requirements could pre-determine summary quality—a short outline of a story might be confusing and unclear but a page long discussion might give an excellent overview of the same issue.

Even systems that perform well on average produce summaries of poor quality for some inputs. For this reason, understanding what aspects of the input make it difficult for summarization becomes an interesting and important issue that has not been addressed in the summarization community until now.

In information retrieval, for example, the variable system performance has been recognized as a research challenge and numerous studies on identifying query difficulty have been carried out (most recently (Cronen-Townsend et al., 2002; Yom-Tov et al., 2005; Carmel et al., 2006)).

In this paper we present results supporting the hypotheses that input topicality cohesiveness and summary length are among the factors that determine summary quality regardless of the choice of summarization strategy (Section 2). The data used for the analyses comes from the annual Document Understanding Conference (DUC) in which various summarization approaches are evaluated on common data, with new test sets provided each year.

In later sections we define a suite of features capturing aspects of the topicality cohesiveness of the input (Section 3) and relate these to system performance, identifying reliable correlates of input difficulty (Section 4). Finally, in Section 5, we demonstrate that the features can be used to build a classifier predicting summarization input difficulty with accuracy considerably above chance level.

## 2 Preliminary analysis and distinctions: DUC 2001

Generic multi-document summarization was featured as a task at the Document Understanding Conference (DUC) in four years, 2001 through 2004. In our study we use the DUC 2001 multi-document task submissions as development data for in-depth analysis and feature selection. There were 29 input sets and 12 automatic summarizers participating in the evaluation that year. Summaries of different

lengths were produced by each system: 50, 100, 200 and 400 words. Each summary was manually evaluated to determine the extent to which its content overlapped with that of a human model, giving a *coverage score*. The content comparison was performed on a subsentence level and was based on elementary discourse units in the model summary.<sup>1</sup>

The coverage scores are taken as an indicator of difficulty of the input: systems achieve low coverage for difficult sets and higher coverage for easy sets. Since we are interested in identifying characteristics of generally difficult inputs rather than in discovering what types of inputs might be difficult for one given system, we use the average system score per set as indicator of general difficulty.

## 2.1 Analysis of variance

Before attempting to derive characteristics of inputs difficult for summarization, we first confirm that indeed expected performance is influenced by the input itself. We performed analysis of variance for DUC 2001 data, with *automatic system* coverage score as the dependent variable, to gain some insight into the factors related to summarization difficulty. The results of the ANOVA with input set, summarizer identity and summary length as factors, as well as the interaction between these, are shown in Table 1.

As expected, summarizer identity is a significant factor: some summarization strategies/systems are more effective than others and produce summaries with higher coverage score. More interestingly, the input set and summary length factors are also highly significant and explain more of the variability in coverage scores than summarizer identity does, as indicated by the larger values of the  $F$  statistic.

**Length** The average automatic summarizer coverage scores increase steadily as length requirements are relaxed, going up from 0.50 for 50-word summaries to 0.76 for 400-word summaries as shown in Table 2 (second row). The general trend we observe is that on average systems are better at producing summaries when more space is available. The dif-

<sup>1</sup>The routinely used tool for automatic evaluation ROUGE was adopted exactly because it was demonstrated it is highly correlated with the manual DUC coverage scores (Lin and Hovy, 2003a; Lin, 2004).

Type	50	100	200	400
Human	1.00	1.17	1.38	1.29
Automatic	0.50	0.55	0.70	0.76
Baseline	0.41	0.46	0.52	0.57

Table 2: Average human, system and baseline coverage scores for different summary lengths of  $N$  words.  $N = 50, 100, 200,$  and  $400$ .

ferences are statistically significant<sup>2</sup> only between 50-word and 200- and 400-word summaries and between 100-word and 400-word summaries. The fact that summary quality improves with increasing summary length has been observed in prior studies as well (Radev and Tam, 2003; Lin and Hovy, 2003b; Kolluru and Gotoh, 2005) but generally little attention has been paid to this fact in system development and no specific user studies are available to show what summary length might be most suitable for specific applications. In later editions of the DUC conference, only summaries of 100 words were produced, focusing development efforts on one of the more demanding length restrictions. The interaction between summary length and summarizer is small but significant (Table 1), with certain summarization strategies more successful at particular summary lengths than at others.

Improved performance as measured by increase in coverage scores is observed for human summarizers as well (shown in the first row of Table 2). Even the baseline systems (first  $n$  words of the most recent article in the input or first sentences from different input articles) show improvement when longer summaries are allowed (performance shown in the third row of the table). It is important to notice that the difference between automatic system and baseline performance increases as the summary length increases—the difference between systems and baselines coverage scores is around 0.1 for the shorter 50- and 100-word summaries but 0.2 for the longer summaries. This fact has favorable implications for practical system developments because it indicates that in applications where somewhat longer summaries are appropriate, automatically produced summaries will be much more informative than a baseline summary.

<sup>2</sup>One-sided t-test, 95% level of significance.

Factor	DF	Sum of squares	Expected mean squares	F stat	Pr(> F)
input	28	150.702	5.382	59.4227	0
summarizer	11	34.316	3.120	34.4429	0
length	3	16.082	5.361	59.1852	0
input:summarizer	306	65.492	0.214	2.3630	0
input:length	84	36.276	0.432	4.7680	0
summarizer:length	33	6.810	0.206	2.2784	0

Table 1: Analysis of variance for coverage scores of automatic systems with input, summarizer, and length as factors.

**Input** The input set itself is a highly significant factor that influences the coverage scores that systems obtain: some inputs are handled by the systems better than others. Moreover, the input interacts both with the summarizers and the summary length.

This is an important finding for several reasons. First, in system evaluations such as DUC the inputs for summarization are manually selected by annotators. There is no specific attempt to ensure that the inputs across different years have on average the same difficulty. Simply assuming this to be the case could be misleading: it is possible in a given year to have “easier” input test set compared to a previous year. Then system performance across years cannot be meaningfully compared, and higher system scores would not be indicative of system improvement between the evaluations.

Second, in summarization applications there is some control over the input for summarization. For example, related documents that need to be summarized could be split into smaller subsets that are more amenable to summarization or routed to an appropriate summarization system than can handle this kind of input using a different strategy, as done for instance in (McKeown et al., 2002).

Because of these important implications we investigate input characteristics and define various features distinguishing easy inputs from difficult ones.

## 2.2 Difficulty for people and machines

Before proceeding to the analysis of input difficulty in multi-document summarization, it is worth mentioning that our study is primarily motivated by system development needs and consequently the focus is on finding out what inputs are easy or difficult *for automatic systems*. Different factors might make summarization difficult *for people*. In order to see to what extent the notion of summarization input dif-

summary length	correlation
50	0.50
100	0.57*
200	0.77**
400	0.70**

Table 3: Pearson correlation between average human and system coverage scores on the DUC 2001 dataset. Significance levels: \* $p < 0.05$  and \*\* $p < 0.00001$ .

iculty is shared between machines and people, we computed the correlation between the average system and average human coverage score at a given summary length for all DUC 2001 test sets (shown in Table 3). The correlation is highest for 200-word summaries, 0.77, which is also highly significant. For shorter summaries the correlation between human and system performance is not significant.

In the remaining part of the paper we deal exclusively with difficulty as defined by system performance, which differs from difficulty for people summarizing the same material as evidenced by the correlations in Table 3. We do not attempt to draw conclusions about any cognitively relevant factors involved in summarizing.

## 2.3 Type of summary and difficulty

In DUC 2001, annotators prepared test sets from five possible predefined input categories:<sup>3</sup>.

**Single event** (3 sets) Documents describing a single event over a timeline (e.g. The Exxon Valdez oil spill).

<sup>3</sup>Participants in the evaluation were aware of the different categories of input and indeed some groups developed systems that handled different types of input employing different strategies (McKeown et al., 2001). In later years, the idea of multi-strategy summarization has been further explored by (Lacatusu et al., 2006)

**Subject** (6 sets) Documents discussing a single topic (e.g. Mad cow disease)

**Biographical** (2 sets) All documents in the input provide information about the same person (e.g. Elizabeth Taylor)

**Multiple distinct events** (12 sets) The documents discuss different events of the same type (e.g. different occasions of police misconduct).

**Opinion** (6 sets) Each document describes a different perspective to a common topic (e.g. views of the senate, congress, public, lawyers etc on the decision by the senate to count illegal aliens in the 1990 census).

Figure 1 shows the average system coverage score for the different input types. The more topically cohesive input types such as *biographical*, *single event* and *subject*, which are more focused on a single entity or news item and narrower in scope, are easier for systems. The average system coverage score for them is higher than for the non-cohesive sets such as multiple distinct events and opinion sets, regardless of summary length. The difference is even more apparently clear when the scores are plotted after grouping input types into cohesive (biographical, single event and subject) and non-cohesive (multiple events and opinion). Such grouping also gives the necessary power to perform statistical test for significance, confirming the difference in coverage scores for the two groups. This is not surprising: a summary of documents describing multiple distinct events of the same type is likely to require higher degree of generalization and abstraction. Summarizing opinions would in addition be highly subjective. A summary of a cohesive set meanwhile would contain facts directly from the input and it would be easier to determine which information is important. The example human summaries for set D32 (single event) and set D19 (opinions) shown below give an idea of the potential difficulties automatic summarizers have to deal with.

**set D32** On 24 March 1989, the oil tanker Exxon Valdez ran aground on a reef near Valdez, Alaska, spilling 8.4 million gallons of crude oil into Prince William Sound. In two days, the oil spread over 100 miles with a heavy toll on wildlife. Cleanup proceeded at a slow pace, and a plan for cleaning 364 miles of Alaskan coastline was released. In June, the tanker was refloated. By early 1990, only 5 to 9 percent of spilled oil was recovered. A federal jury indicted Exxon

on five criminal charges and the Valdez skipper was guilty of negligent discharge of oil.

**set D19** Congress is debating whether or not to count illegal aliens in the 1990 census. Congressional House seats are apportioned to the states and huge sums of federal money are allocated based on census population. California, with an estimated half of all illegal aliens, will be greatly affected. *Those arguing for inclusion say that the Constitution does not mention “citizens”, but rather, instructs that House apportionment be based on the “whole number of persons” residing in the various states. Those opposed say that the framers were unaware of this issue. “Illegal aliens” did not exist in the U.S. until restrictive immigration laws were passed in 1875.*

The manual set-type labels give an intuitive idea of what factors might be at play but it is desirable to devise more specific measures to predict difficulty. Do such measures exist? Is there a way to automatically distinguish cohesive (easy) from non-cohesive (difficult) sets? In the next section we define a number of features that aim to capture the cohesiveness of an input set and show that some of them are indeed significantly related to set difficulty.

### 3 Features

We implemented 14 features for our analysis of input set difficulty. The working hypothesis is that cohesive sets with clear topics are easier to summarize and the features we define are designed to capture aspects of input cohesiveness.

**Number of sentences** in the input, calculated over all articles in the input set. Shorter inputs should be easier as there will be less information loss between the summary and the original material.

**Vocabulary size** of the input set, equal to the number of unique words in the input. Smaller vocabularies would be characteristic of easier sets.

**Percentage of words used only once** in the input. The rationale behind this feature is that cohesive input sets contain news articles dealing with a clearly defined topic, so words will be reused across documents. Sets that cover disparate events and opinions are likely to contain more words that appear in the input only once.

**Type-token ratio** is a measure of the lexical variation in an input set and is equal to the input vocabulary size divided by the number of words in the



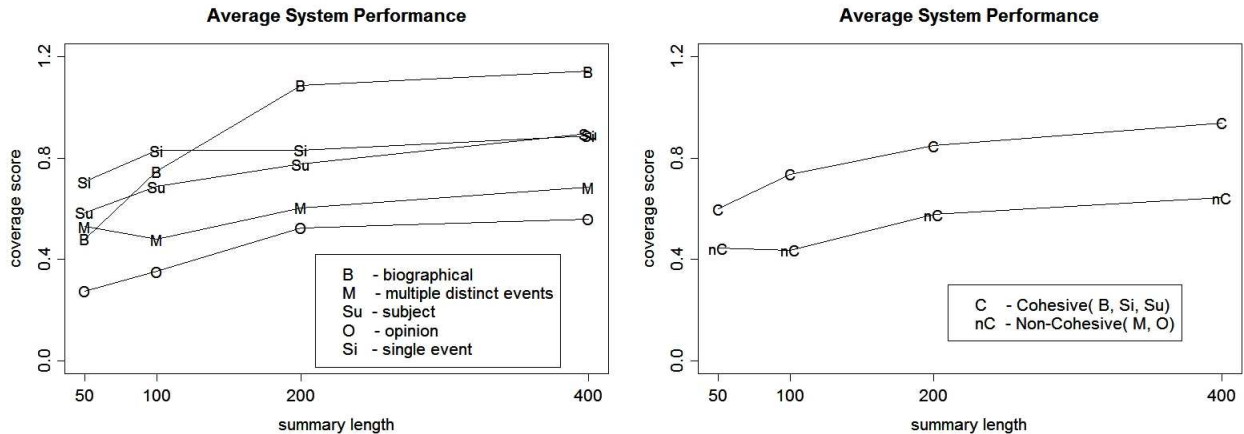


Figure 1: Average system coverage scores for summaries in a category

input. A high type-token ratio indicates there is little (lexical) repetition in the input, a possible side-effect of non-cohesiveness.

**Entropy** of the input set. Let  $X$  be a discrete random variable taking values from the finite set  $V = \{w_1, \dots, w_n\}$  where  $V$  is the vocabulary of the input set and  $w_i$  are the words that appear in the input. The probability distribution  $p(w) = Pr(X = w)$  can be easily calculated using frequency counts from the input. The entropy of the input set is equal to the entropy of  $X$ :

$$H(X) = - \sum_{i=1}^{i=n} p(w_i) \log_2 p(w_i) \quad (1)$$

**Average, minimum and maximum cosine overlap** between the news articles in the input. Repetition in the input is often exploited as an indicator of importance by different summarization approaches (Luhn, 1958; Barzilay et al., 1999; Radev et al., 2004; Nenkova et al., 2006). The more similar the different documents in the input are to each other, the more likely there is repetition across documents at various granularities.

Cosine similarity between the document vector representations is probably the easiest and most commonly used among the various similarity measures. We use tf\*idf weights in the vector representations, with term frequency (tf) normalized by the total number of words in the document in order to remove bias resulting from high frequencies by virtue of higher document length alone.

The cosine similarity between two (document representation) vectors  $v_1$  and  $v_2$  is given by  $\cos\theta = \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|}$ . A value of 0 indicates that the vectors are orthogonal and dissimilar, a value of 1 indicates perfectly similar documents in terms of the words contained in them.

To compute the cosine overlap features, we find the pairwise cosine similarity between each two documents in an input set and compute their average. The minimum and maximum overlap features are also computed as an indication of the overlap bounds. We expect cohesive inputs to be composed of similar documents, hence the cosine overlaps in these sets of documents must be higher than those in non-cohesive inputs.

**KL divergence** Another measure of relatedness of the documents comprising an input set is the difference in word distributions in the input compared to the word distribution in a large collection of diverse texts. If the input is found to be largely different from a generic collection, it is plausible to assume that the input is not a random collection of articles but rather is defined by a clear topic discussed within and across the articles. It is reasonable to expect that the higher the divergence is, the easier it is to define what is important in the article and hence the easier it is to produce a good summary.

For computing the distribution of words in a general background corpus, we used all the inputs sets from DUC years 2001 to 2006. The divergence measure we used is the Kullback Leibler divergence, or

relative entropy, between the input ( $I$ ) and collection language models. Let  $p_{inp}(w)$  be the probability of the word  $w$  in the input and  $p_{coll}(w)$  be the probability of the word occurring in the large background collection. Then the relative entropy between the input and the collection is given by

$$\text{KL divergence} = \sum_{w \in I} p_{inp}(w) \log_2 \frac{p_{inp}(w)}{p_{coll}(w)} \quad (2)$$

Low KL divergence from a random background collection may be characteristic of highly non-cohesive inputs consisting of unrelated documents.

**Number of topic signature terms** for the input set. The idea of topic signature terms was introduced by Lin and Hovy (Lin and Hovy, 2000) in the context of single document summarization, and was later used in several multi-document summarization systems (Conroy et al., 2006; Lacatusu et al., 2004; Gupta et al., 2007).

Lin and Hovy’s idea was to automatically identify words that are descriptive for a cluster of documents on the same topic, such as the input to a multi-document summarizer. We will call this cluster  $T$ . Since the goal is to find descriptive terms for the cluster, a comparison collection of documents not on the topic is also necessary (we will call this background collection  $NT$ ).

Given  $T$  and  $NT$ , the likelihood ratio statistic (Dunning, 1994) is used to identify the topic signature terms. The probabilistic model of the data allows for statistical inference in order to decide which terms  $t$  are associated with  $T$  more strongly than with  $NT$  than one would expect by chance.

More specifically, there are two possibilities for the distribution of a term  $t$ : either it is very indicative of the topic of cluster  $T$ , and appears more often in  $T$  than in documents from  $NT$ , or the term  $t$  is not topical and appears with equal frequency across both  $T$  and  $NT$ . These two alternatives can be formally written as the following hypotheses:

H1:  $P(t|T) = P(t|NT) = p$  ( $t$  is not a descriptive term for the input)

H2:  $P(t|T) = p_1$  and  $P(t|NT) = p_2$  and  $p_1 > p_2$  ( $t$  is a descriptive term)

In order to compute the likelihood of each hypothesis given the collection of the background docu-

ments and the topic cluster, we view them as a sequence of words  $w_i: w_1 w_2 \dots w_N$ . The occurrence of a given word  $t$ ,  $w_i = t$ , can thus be viewed a Bernoulli trial with probability  $p$  of success, with success occurring when  $w_i = t$  and failure otherwise.

The probability of observing the term  $t$  appearing  $k$  times in  $N$  trials is given by the binomial distribution

$$b(k, N, p) = \binom{N}{k} p^k (1-p)^{N-k} \quad (3)$$

We can now compute

$$\lambda = \frac{\text{Likelihood of the data given H1}}{\text{Likelihood of the data given H2}} \quad (4)$$

which is equal to

$$\lambda = \frac{b(c_t, N, p)}{b(c_T, N_T, p_1) * b(c_{NT}, N_{NT}, p_2)} \quad (5)$$

The maximum likelihood estimates for the probabilities can be computed directly.  $p = \frac{c_t}{N}$ , where  $c_t$  is equal to the number of times term  $t$  appeared in the entire corpus  $T+NT$ , and  $N$  is the number of words in the entire corpus. Similarly,  $p_1 = \frac{c_T}{N_T}$ , where  $c_T$  is the number of times term  $t$  occurred in  $T$  and  $N_T$  is the number of all words in  $T$ .  $p_2 = \frac{c_{NT}}{N_{NT}}$ , where  $c_{NT}$  is the number of times term  $t$  occurred in  $NT$  and  $N_{NT}$  is the total number of words in  $NT$ .

$-2 \log \lambda$  has a well-know distribution:  $\chi^2$ . Bigger values of  $-2 \log \lambda$  indicate that the likelihood of the data under H2 is higher, and the  $\chi^2$  distribution can be used to determine when it is significantly higher ( $-2 \log \lambda$  exceeding 10 gives a significance level of 0.001 and is the cut-off we used).

For terms for which the computed  $-2 \log \lambda$  is higher than 10, we can infer that they occur more often with the topic  $T$  than in a general corpus  $NT$ , and we can dub them “topic signature terms”.

#### Percentage of signature terms in vocabulary

The number of signature terms gives the total count of topic signatures over all the documents in the input. However, the number of documents in an input set and the size of the individual documents across different sets are not the same. It is therefore possible that the mere count feature is biased to the length

and number of documents in the input set. To account for this, we add the percentage of topic words in the vocabulary as a feature.

**Average, minimum and maximum topic signature overlap** between the documents in the input. Cosine similarity measures the overlap between two documents based on all the words appearing in them. A more refined document representation can be defined by assuming the document vectors contain only the topic signature words rather than all words. A high overlap of topic words across two documents is indicative of shared topicality. The average, minimum and maximum pairwise cosine overlap between the tf\*idf weighted topic signature vectors of the two documents are used as features for predicting input cohesiveness. If the overlap is large, then the topic is similar across the two documents and hence their combination will yield a cohesive input.

#### 4 Feature selection

Table 4 shows the results from a one-sided t-test comparing the values of the various features for the easy and difficult input set classes. The comparisons are for summary length of 100 words because in later years only such summaries were evaluated. The binary easy/difficult classes were assigned based on the average system coverage score for the given set, with half of the sets assigned to each class.

In addition to the t-tests we also calculated Pearson’s correlation (shown in Table 5) between the features and the average system coverage score for each set. In the correlation analysis the input sets are not classified into easy or difficult but rather the real valued coverage scores are used directly. Overall, the features that were identified by the t-test as most descriptive of the differences between easy and difficult inputs were also the ones with higher correlations with real-valued coverage scores.

Our expectations in defining the features are confirmed by the correlation results. For example, systems have low coverage scores for sets with high-entropy vocabularies as indicated by the negative and high by absolute value correlation (-0.4256). Sets with high entropy are those in which there is little repetition within and across different articles, and for which it is subsequently difficult to deter-

feature	t-stat	p-value
KL divergence*	-2.4725	0.01
% of sig. terms in vocab*	-2.0956	0.02
average cosine overlap*	-2.1227	0.02
vocabulary size*	1.9378	0.03
set entropy*	2.0288	0.03
average sig. term overlap*	-1.8803	0.04
max cosine overlap	-1.6968	0.05
max topic signature overlap	-1.6380	0.06
number of sentences	1.4780	0.08
min topic signature overlap	-0.9540	0.17
number of signature terms	0.8057	0.21
min cosine overlap	-0.2654	0.39
% of words used only once	0.2497	0.40
type-token ratio	0.2343	0.41

\*Significant at a 95% confidence level( $p < 0.05$ )

Table 4: Comparison of non-cohesive (average system coverage score < median average system score) vs cohesive sets for summary length of 100 words

mine what is the most important content. On the other hand, sets characterized by bigger KL divergence are easier—there the distribution of words is skewed compared to a general collection of articles, with important topic words occurring more often.

Easy to summarize sets are characterized by low entropy, small vocabulary, high average cosine and average topic signature overlaps, high KL divergence and a high percentage of the vocabulary consists of topic signature terms.

#### 5 Classification results

We used the 192 sets from multi-document summarization DUC evaluations in 2002 (55 generic sets), 2003 (30 generic summary sets and 7 viewpoint sets) and 2004 (50 generic and 50 biography sets) to train and test a logistic regression classifier. The sets from all years were pooled together and evenly divided into easy and difficult inputs based on the average system coverage score for each set.

Table 6 shows the results from 10-fold cross validation. SIG is a classifier based on the six features identified as significant in distinguishing easy from difficult inputs based on a t-test comparison (Table 4). SIG+yt has two additional features: the year and the type of summarization input (generic, viewpoint and biographical). ALL is a classifier based on all 14 features defined in the previous section, and

feature	correlation
set entropy	-0.4256
KL divergence	0.3663
vocabulary size	-0.3610
% of sig. terms in vocab	0.3277
average sig. term overlap	0.2860
number of sentences	-0.2511
max topic signature overlap	0.2416
average cosine overlap	0.2244
number of signature terms	-0.1880
max cosine overlap	0.1337
min topic signature overlap	0.0401
min cosine overlap	0.0308
type-token ratio	-0.0276
% of words used only once	-0.0025

Table 5: Correlation between coverage score and feature values for the 29 DUC'01 100-word summaries.

features	accuracy	P	R	F
SIG	56.25%	0.553	0.600	0.576
SIG+yt	69.27%	0.696	0.674	0.684
ALL	61.45%	0.615	0.589	0.600
ALL+yt	65.10%	0.643	0.663	0.653

Table 6: Logistic regression classification results (accuracy, precision, recall and f-measure) for balanced data of 100-word summaries from DUC'02 through DUC'04.

ALL+yt also includes the year and task features.

Classification accuracy is considerably higher than the 50% random baseline. Using all features yields better accuracy (61%) than using solely the 6 significant features (accuracy of 56%). In both cases, adding the year and task leads to extra 3% net improvement. The best overall results are for the SIG+yt classifier with net improvement over the baseline equal to 20%. At the same time, it should be taken into consideration that the amount of training data for our experiments is small: a total of 192 sets. Despite this, the measures of input cohesiveness capture enough information to result in a classifier with above-baseline performance.

## 6 Conclusions

We have addressed the question of what makes the writing of a summary for a multi-document input difficult. Summary length is a significant factor, with all summarizers (people, machines and baselines) performing better at longer summary lengths.

An exploratory analysis of DUC 2001 indicated that systems produce better summaries for cohesive inputs dealing with a clear topic (single event, subject and biographical sets) while non-cohesive sets about multiple events and opposing opinions are consistently of lower quality. We defined a number of features aimed at capturing input cohesiveness, ranging from simple features such as input length and size to more sophisticated measures such as input set entropy, KL divergence from a background corpus and topic signature terms based on log-likelihood ratio.

Generally, easy to summarize sets are characterized by low entropy, small vocabulary, high average cosine and average topic signature overlaps, high KL divergence and a high percentage of the vocabulary consists of topic signature terms. Experiments with a logistic regression classifier based on the features further confirms that input cohesiveness is predictive of the difficulty it will pose to automatic summarizers.

Several important notes can be made. First, it is important to develop strategies that can better handle non-cohesive inputs, reducing fluctuations in system performance. Most current systems are developed with the expectation they can handle any input but this is evidently not the case and more attention should be paid to the issue. Second, the interpretations of year to year evaluations can be affected. As demonstrated, the properties of the input have a considerable influence on summarization quality. If special care is not taken to ensure that the difficulty of inputs in different evaluations is kept more or less the same, results from the evaluations are not comparable and we cannot make general claims about progress and system improvements between evaluations. Finally, the presented results are clearly just a beginning in understanding of summarization difficulty. A more complete characterization of summarization input will be necessary in the future.

## References

- Regina Barzilay, Kathleen McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.
- David Carmel, Elad Yom-Tov, Adam Darlow, and Dan

- Pelleg. 2006. What makes a query difficult? In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 390–397.
- John Conroy, Judith Schlesinger, and Dianne O’Leary. 2006. Topic-focused multi-document summarization using an approximate oracle score. In *Proceedings of ACL, companion volume*.
- Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting query performance. In *Proceedings of the 25th Annual International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 299–306.
- Ted Dunning. 1994. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.
- Surabhi Gupta, Ani Nenkova, and Dan Jurafsky. 2007. Measuring importance and query relevance in topic-focused multi-document summarization. In *ACL’07, companion volume*.
- BalaKrishna Kolluru and Yoshihiko Gotoh. 2005. On the subjectivity of human authored short summaries. In *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.
- Finley Lacatusu, Andrew Hickl, Sanda Harabagiu, and Luke Nezdá. 2004. Lite\_gistexter at duc2004. In *Proceedings of the 4th Document Understanding Conference (DUC’04)*.
- F. Lacatusu, A. Hickl, K. Roberts, Y. Shi, J. Bensley, B. Rink, P. Wang, and L. Taylor. 2006. Lcc’s gistexter at duc 2006: Multi-strategy multi-document summarization. In *DUC’06*.
- Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics*, pages 495–501.
- Chin-Yew Lin and Eduard Hovy. 2003a. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT-NAACL 2003*.
- Chin-Yew Lin and Eduard Hovy. 2003b. The potential and limitations of automatic sentence extraction for summarization. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop*, pages 73–80.
- Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *ACL Text Summarization Workshop*.
- H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.
- K. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, B. Schiffman, and S. Teufel. 2001. Columbia multi-document summarization: Approach and evaluation. In *DUC’01*.
- Kathleen McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. 2002. Tracking and summarizing news on a daily basis with columbia’s newsblaster. In *Proceedings of the 2nd Human Language Technologies Conference HLT-02*.
- Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of SIGIR*.
- Dragomir Radev and Daniel Tam. 2003. Single-document and multi-document summary evaluation via relative utility. In *Poster session, International Conference on Information and Knowledge Management (CIKM’03)*.
- Dragomir Radev, Hongyan Jing, Malgorzata Sty, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40:919–938.
- Elad Yom-Tov, Shai Fine, David Carmel, and Adam Darlow. 2005. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *SIGIR ’05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 512–519.

# You talking to me? A Corpus and Algorithm for Conversation Disentanglement

Micha Elsner and Eugene Charniak

Brown Laboratory for Linguistic Information Processing (BLLIP)

Brown University

Providence, RI 02912

{melsner, ec}@cs.brown.edu

## Abstract

When multiple conversations occur simultaneously, a listener must decide which conversation each utterance is part of in order to interpret and respond to it appropriately. We refer to this task as disentanglement. We present a corpus of Internet Relay Chat (IRC) dialogue in which the various conversations have been manually disentangled, and evaluate annotator reliability. This is, to our knowledge, the first such corpus for internet chat. We propose a graph-theoretic model for disentanglement, using discourse-based features which have not been previously applied to this task. The model's predicted disentanglements are highly correlated with manual annotations.

## 1 Motivation

Simultaneous conversations seem to arise naturally in both informal social interactions and multi-party typed chat. Aoki et al. (2006)'s study of voice conversations among 8-10 people found an average of 1.76 conversations (floors) active at a time, and a maximum of four. In our chat corpus, the average is even higher, at 2.75. The typical conversation, therefore, is one which is interrupted— frequently.

Disentanglement is the clustering task of dividing a transcript into a set of distinct conversations. It is an essential prerequisite for any kind of higher-level dialogue analysis: for instance, consider the multi-party exchange in figure 1.

Contextually, it is clear that this corresponds to two conversations, and Felicia's<sup>1</sup> response “excel-

**(Chanel)** Felicia: google works :)  
**(Gale)** Arlie: you guys have never worked in a factory before have you  
**(Gale)** Arlie: there's some real unethical stuff that goes on  
**(Regine)** hands Chanel a trophy  
**(Arlie)** Gale, of course ... thats how they make money  
**(Gale)** and people lose limbs or get killed  
**(Felicia)** excellent

Figure 1: Some (abridged) conversation from our corpus.

lent” is intended for Chanel and Regine. A straightforward reading of the transcript, however, might interpret it as a response to Gale's statement immediately preceding.

Humans are adept at disentanglement, even in complicated environments like crowded cocktail parties or chat rooms; in order to perform this task, they must maintain a complex mental representation of the ongoing discourse. Moreover, they adapt their utterances to some degree to make the task easier (O'Neill and Martin, 2003), which suggests that disentanglement is in some sense a “difficult” discourse task.

Disentanglement has two practical applications. One is the analysis of pre-recorded transcripts in order to extract some kind of information, such as question-answer pairs or summaries. These tasks should probably take as input each separate conversation, rather than the entire transcript. Another

identifiers for ethical reasons.

<sup>1</sup>Real user nicknames are replaced with randomly selected

application is as part of a user-interface system for active participants in the chat, in which users target a conversation of interest which is then highlighted for them. Aoki et al. (2003) created such a system for speech, which users generally preferred to a conventional system—when the disentanglement worked!

Previous attempts to solve the problem (Aoki et al., 2006; Aoki et al., 2003; Camtepe et al., 2005; Acar et al., 2005) have several flaws. They cluster speakers, not utterances, and so fail when speakers move from one conversation to another. Their features are mostly time gaps between one utterance and another, without effective use of utterance content. Moreover, there is no framework for a principled comparison of results: there are no reliable annotation schemes, no standard corpora, and no agreed-upon metrics.

We attempt to remedy these problems. We present a new corpus of manually annotated chat room data and evaluate annotator reliability. We give a set of metrics describing structural similarity both locally and globally. We propose a model which uses discourse structure and utterance contents in addition to time gaps. It partitions a chat transcript into distinct conversations, and its output is highly correlated with human annotations.

## 2 Related Work

Two threads of research are direct attempts to solve the disentanglement problem: Aoki et al. (2006), Aoki et al. (2003) for speech and Camtepe et al. (2005), Acar et al. (2005) for chat. We discuss their approaches below. However, we should emphasize that we cannot compare our results directly with theirs, because none of these studies publish results on human-annotated data. Although Aoki et al. (2006) construct an annotated speech corpus, they give no results for model performance, only user satisfaction with their conversational system. Camtepe et al. (2005) and Acar et al. (2005) do give performance results, but only on synthetic data.

All of the previous approaches treat the problem as one of clustering speakers, rather than utterances. That is, they assume that during the window over which the system operates, a particular speaker is engaging in only one conversation. Camtepe et al. (2005) assume this is true throughout the entire tran-

script; real speakers, by contrast, often participate in many conversations, sequentially or sometimes even simultaneously. Aoki et al. (2003) analyze each thirty-second segment of the transcript separately. This makes the single-conversation restriction somewhat less severe, but has the disadvantage of ignoring all events which occur outside the segment.

Acar et al. (2005) attempt to deal with this problem by using a fuzzy algorithm to cluster speakers; this assigns each speaker a distribution over conversations rather than a hard assignment. However, the algorithm still deals with speakers rather than utterances, and cannot determine which conversation any particular utterance is part of.

Another problem with these approaches is the information used for clustering. Aoki et al. (2003) and Camtepe et al. (2005) detect the arrival times of messages, and use them to construct an affinity graph between participants by detecting turn-taking behavior among pairs of speakers. (Turn-taking is typified by short pauses between utterances; speakers aim neither to interrupt nor leave long gaps.) Aoki et al. (2006) find that turn-taking on its own is inadequate. They motivate a richer feature set, which, however, does not yet appear to be implemented. Acar et al. (2005) adds word repetition to their feature set. However, their approach deals with all word repetitions on an equal basis, and so degrades quickly in the presence of *noise words* (their term for words which shared across conversations) to almost complete failure when only 1/2 of the words are shared.

To motivate our own approach, we examine some linguistic studies of discourse, especially analysis of multi-party conversation. O’Neill and Martin (2003) point out several ways in which multi-party text chat differs from typical two-party conversation. One key difference is the frequency with which participants mention each others’ names. They hypothesize that mentioning is a strategy which participants use to make disentanglement easier, compensating for the lack of cues normally present in face-to-face dialogue. Mentions (such as Gale’s comments to Arlie in figure 1) are very common in our corpus, occurring in 36% of comments, and provide a useful feature.

Another key difference is that participants may create a new conversation (floor) at any time, a process which Sacks et al. (1974) calls *schisming*. Dur-

ing a schism, a new conversation is formed, not necessarily because of a shift in the topic, but because certain participants have refocused their attention onto each other, and away from whoever held the floor in the parent conversation.

Despite these differences, there are still strong similarities between chat and other conversations such as meetings. Our feature set incorporates information which has proven useful in meeting segmentation (Galley et al., 2003) and the task of detecting addressees of a specific utterance in a meeting (Jovanovic et al., 2006). These include word repetitions, utterance topic, and *cue words* which can indicate the bounds of a segment.

### 3 Dataset

Our dataset is recorded from the IRC (Internet Relay Chat) channel `##LINUX` at *freenode.net*, using the freely-available *gaim* client. `##LINUX` is an unofficial tech support line for the Linux operating system, selected because it is one of the most active chat rooms on *freenode*, leading to many simultaneous conversations, and because its content is typically inoffensive. Although it is notionally intended only for tech support, it includes large amounts of social chat as well, such as the conversation about factory work in the example above (figure 1).

The entire dataset contains 52:18 hours of chat, but we devote most of our attention to three annotated sections: development (706 utterances; 2:06 hr) and test (800 utts.; 1:39 hr) plus a short pilot section on which we tested our annotation system (359 utts.; 0:58 hr).

#### 3.1 Annotation

Our annotators were seven university students with at least some familiarity with the Linux OS, although in some cases very slight. Annotation of the test dataset typically took them about two hours. In all, we produced six annotations of the test set<sup>2</sup>.

Our annotation scheme marks each utterance as part of a single conversation. Annotators are instructed to create as many, or as few conversations as they need to describe the data. Our instructions state that a conversation can be between any number of

---

<sup>2</sup>One additional annotation was discarded because the annotator misunderstood the task.

people, and that, “We mean conversation in the typical sense: a discussion in which the participants are all reacting and paying attention to one another. . . it should be clear that the comments inside a conversation fit together.” The annotation system itself is a simple Java program with a graphical interface, intended to appear somewhat similar to a typical chat client. Each speaker’s name is displayed in a different color, and the system displays the elapsed time between comments, marking especially long pauses in red. Annotators group sentences into conversations by clicking and dragging them onto each other.

#### 3.2 Metrics

Before discussing the annotations themselves, we will describe the metrics we use to compare different annotations; these measure both how much our annotators agree with each other, and how well our model and various baselines perform. Comparing clusterings with different numbers of clusters is a non-trivial task, and metrics for agreement on supervised classification, such as the  $\kappa$  statistic, are not applicable.

To measure global similarity between annotations, we use *one-to-one accuracy*. This measure describes how well we can extract whole conversations intact, as required for summarization or information extraction. To compute it, we pair up conversations from the two annotations to maximize the total overlap<sup>3</sup>, then report the percentage of overlap found.

If we intend to monitor or participate in the conversation as it occurs, we will care more about local judgements. The *local agreement* metric counts agreements and disagreements within a context  $k$ . We consider a particular utterance: the previous  $k$  utterances are each in either the *same* or a *different* conversation. The  $loc_k$  score between two annotators is their average agreement on these  $k$  same/different judgements, averaged over all utterances. For example,  $loc_1$  counts pairs of adjacent utterances for which two annotations agree.



	Mean	Max	Min
Conversations	81.33	128	50
Avg. Conv. Length	10.6	16.0	6.2
Avg. Conv. Density	2.75	2.92	2.53
Entropy	4.83	6.18	3.00
1-to-1	52.98	63.50	35.63
$loc_3$	81.09	86.53	74.75
M-to-1 (by entropy)	86.70	94.13	75.50

Table 1: Statistics on 6 annotations of 800 lines of chat transcript. Inter-annotator agreement metrics (below the line) are calculated between distinct pairs of annotations.

### 3.3 Discussion

A statistical examination of our data (table 1) shows that that it is eminently suitable for disentanglement: the average number of conversations active at a time is 2.75. Our annotators have high agreement on the local metric (average of 81.1%). On the 1-to-1 metric, they disagree more, with a mean overlap of 53.0% and a maximum of only 63.5%. This level of overlap does indicate a useful degree of reliability, which cannot be achieved with naive heuristics (see section 5). Thus measuring 1-to-1 overlap with our annotations is a reasonable evaluation for computational models. However, we feel that the major source of disagreement is one that can be remedied in future annotation schemes: the specificity of the individual annotations.

To measure the level of detail in an annotation, we use the information-theoretic entropy of the random variable which indicates which conversation an utterance is in. This quantity is non-negative, increasing as the number of conversations grow and their size becomes more balanced. It reaches its maximum, 9.64 bits for this dataset, when each utterance is placed in a separate conversation. In our annotations, it ranges from 3.0 to 6.2. This large variation shows that some annotators are more specific than others, but does not indicate how much they agree on the general structure. To measure this, we introduce the many-to-one accuracy. This measurement is asymmetrical, and maps each of the conversations of the *source* annotation to the single con-

<sup>3</sup>This is an example of max-weight bipartite matching, and can be computed optimally using, eg, max-flow. The widely used greedy algorithm is a two-approximation, although we have not found large differences in practice.

**(Lai)** need money  
**(Astrid)** suggest a paypal fund or similar  
**(Lai)** Azzie [sic; typo for Astrid?]: my shack guy here said paypal too but i have no local bank acct  
**(Felicia)** second's Azzie's suggestion  
**(Gale)** we should charge the noobs \$1 per question to [Lai's] paypal  
**(Felicia)** bingo!  
**(Gale)** we'd have the money in 2 days max  
**(Azzie)** Lai: hrm, Have you tried to set one up?  
**(Arlie)** the federal reserve system conspiracy is keeping you down man  
**(Felicia)** Gale: all ubuntu users .. pay up!  
**(Gale)** and susers pay double  
**(Azzie)** I certainly would make suse users pay.  
**(Hildegard)** triple.  
**(Lai)** Azzie: not since being offline  
**(Felicia)** it doesn't need to be "in state" either

Figure 2: A schism occurring in our corpus (abridged): not all annotators agree on where the thread about charging for answers to techical questions diverges from the one about setting up Paypal accounts. Either Gale's or Azzie's first comment seems to be the schism-inducing utterance.

versation in the *target* with which it has the greatest overlap, then counts the total percentage of overlap. This is not a statistic to be optimized (indeed, optimization is trivial: simply make each utterance in the source into its own conversation), but it can give us some intuition about specificity. In particular, if one subdivides a coarse-grained annotation to make a more specific variant, the many-to-one accuracy from fine to coarse remains 1. When we map high-entropy annotations (fine) to lower ones (coarse), we find high many-to-one accuracy, with a mean of 86%, which implies that the more specific annotations have mostly the same large-scale boundaries as the coarser ones.

By examining the local metric, we can see even more: local correlations are good, at an average of 81.1%. This means that, in the three-sentence window preceding each sentence, the annotators are of-

ten in agreement. If they recognize subdivisions of a large conversation, these subdivisions tend to be contiguous, not mingled together, which is why they have little impact on the local measure.

We find reasons for the annotators’ disagreement about appropriate levels of detail in the linguistic literature. As mentioned, new conversations often break off from old ones in schisms. Aoki et al. (2006) discuss conversational features associated with schisming and the related process of *affiliation*, by which speakers attach themselves to a conversation. Schisms often branch off from asides or even normal comments (*toss-outs*) within an existing conversation. This means that there is no clear beginning to the new conversation— at the time when it begins, it is not clear that there are two separate floors, and this will not become clear until distinct sets of speakers and patterns of turn-taking are established. Speakers, meanwhile, take time to orient themselves to the new conversation. An example schism is shown in Figure 2.

Our annotation scheme requires annotators to mark each utterance as part of a single conversation, and distinct conversations are not related in any way. If a schism occurs, the annotator is faced with two options: if it seems short, they may view it as a mere digression and label it as part of the parent conversation. If it seems to deserve a place of its own, they will have to separate it from the parent, but this severs the initial comment (an otherwise unremarkable aside) from its context. One or two of the annotators actually remarked that this made the task confusing. Our annotators seem to be either “splitters” or “lumpers”— in other words, each annotator seems to aim for a consistent level of detail, but each one has their own idea of what this level should be.

As a final observation about the dataset, we test the appropriateness of the assumption (used in previous work) that each speaker takes part in only one conversation. In our data, the average speaker takes part in about 3.3 conversations (the actual number varies for each annotator). The more talkative a speaker is, the more conversations they participate in, as shown by a plot of conversations versus utterances (Figure 3). The assumption is not very accurate, especially for speakers with more than 10 utterances.

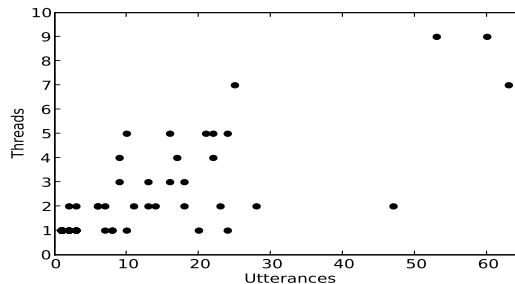


Figure 3: Utterances versus conversations participated in per speaker on development data.

## 4 Model

Our model for disentanglement fits into the general class of graph partitioning algorithms (Roth and Yih, 2004) which have been used for a variety of tasks in NLP, including the related task of meeting segmentation (Malioutov and Barzilay, 2006). These algorithms operate in two stages: first, a binary classifier marks each pair of items as alike or different, and second, a consistent partition is extracted.<sup>4</sup>

### 4.1 Classification

We use a maximum-entropy classifier (Daumé III, 2004) to decide whether a pair of utterances  $x$  and  $y$  are in *same* or *different* conversations. The most likely class is *different*, which occurs 57% of the time in development data. We describe the classifier’s performance in terms of raw accuracy (correct decisions / total), precision and recall of the *same* class, and F-score, the harmonic mean of precision and recall. Our classifier uses several types of features (table 2). The chat-specific features yield the highest accuracy and precision. Discourse and content-based features have poor accuracy on their own (worse than the baseline), since they work best on nearby pairs of utterances, and tend to fail on more distant pairs. Paired with the time gap feature, however, they boost accuracy somewhat and produce substantial gains in recall, encouraging the model to group related utterances together.

The time gap, as discussed above, is the most widely used feature in previous work. We exam-

<sup>4</sup>Our first attempt at this task used a Bayesian generative model. However, we could not define a sharp enough posterior over new sentences, which made the model unstable and overly sensitive to its prior.

**Chat-specific (Acc 73: Prec: 73 Rec: 61 F: 66)**

Time	The time between $x$ and $y$ in seconds, bucketed logarithmically.
Speaker	$x$ and $y$ have the same speaker.
Mention	$x$ mentions $y$ (or vice versa), both mention the same name, either mentions any name.

**Discourse (Acc 52: Prec: 47 Rec: 77 F: 58)**

Cue words	Either $x$ or $y$ uses a greeting (“hello” &c), an answer (“yes”, “no” &c), or thanks.
Question	Either asks a question (explicitly marked with “?”).
Long	Either is long ( $> 10$ words).

**Content (Acc 50: Prec: 45 Rec: 74 F: 56)**

Repeat( $i$ )	The number of words shared between $x$ and $y$ which have unigram probability $i$ , bucketed logarithmically.
Tech	Whether both $x$ and $y$ use technical jargon, neither do, or only one does.

**Combined (Acc 75: Prec: 73 Rec: 68 F: 71)**

Table 2: Feature functions with performance on development data.

ine the distribution of pauses between utterances in the same conversation. Our choice of a logarithmic bucketing scheme is intended to capture two characteristics of the distribution (figure 4). The curve has its maximum at 1-3 seconds, and pauses shorter than a second are less common. This reflects turn-taking behavior among participants; participants in the same conversation prefer to wait for each others’ responses before speaking again. On the other hand, the curve is quite heavy-tailed to the right, leading us to bucket long pauses fairly coarsely.

Our discourse-based features model some pair-

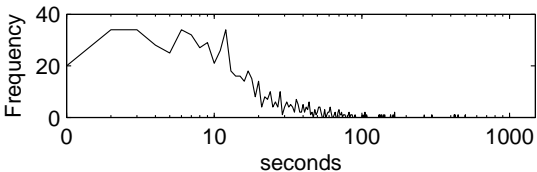


Figure 4: Distribution of pause length (log-scaled) between utterances in the same conversation.

wise relationships: questions followed by answers, short comments reacting to longer ones, greetings at the beginning and thanks at the end.

Word repetition is a key feature in nearly every model for segmentation or coherence, so it is no surprise that it is useful here. We bucket repeated words by their unigram probability<sup>5</sup> (measured over the entire 52 hours of transcript). The bucketing scheme allows us to deal with “noise words” which are repeated coincidentally.

The point of the repetition feature is of course to detect sentences with similar topics. We also find that sentences with technical content are more likely to be related than non-technical sentences. We label an utterance as technical if it contains a web address, a long string of digits, or a term present in a guide for novice Linux users<sup>6</sup> but not in a large news corpus (Graff, 1995)<sup>7</sup>. This is a light-weight way to capture one “semantic dimension” or cluster of related words, in a corpus which is not amenable to full LSA or similar techniques. LSA in text corpora yields a better relatedness measure than simple repetition (Foltz et al., 1998), but is ineffective in our corpus because of its wide variety of topics and lack of distinct document boundaries.

Pairs of utterances which are widely separated in the discourse are unlikely to be directly related—even if they are part of the same conversation, the link between them is probably a long chain of intervening utterances. Thus, if we run our classifier on a pair of very distant utterances, we expect it to default to the majority class, which in this case will be *different*, and this will damage our performance in case the two are really part of the same conversation. To deal with this, we run our classifier only on utterances separated by 129 seconds or less. This is the last of our logarithmic buckets in which the classifier has a significant advantage over the majority baseline. For 99.9% of utterances in an ongoing conversation, the previous utterance in that conversation is within this gap, and so the system has a

<sup>5</sup>We discard the 50 most frequent words entirely.

<sup>6</sup>“Introduction to Linux: A Hands-on Guide”. Machtelt Garrels. Edition 1.25 from <http://tldp.org/LDP/intro-linux/html/intro-linux.html>.

<sup>7</sup>Our data came from the LA times, 94-97— helpfully, it pre-dates the current wide coverage of Linux in the mainstream press.

chance of correctly linking the two.

On test data, the classifier has a mean accuracy of 68.2 (averaged over annotations). The mean precision of *same conversation* is 53.3 and the recall is 71.3, with mean F-score of 60. This error rate is high, but the partitioning procedure allows us to recover from some of the errors, since if nearby utterances are grouped correctly, the bad decisions will be outvoted by good ones.

## 4.2 Partitioning

The next step in the process is to cluster the utterances. We wish to find a set of clusters for which the weighted accuracy of the classifier would be maximal; this is an example of *correlation clustering* (Bansal et al., 2004), which is NP-complete<sup>8</sup>. Finding an exact solution proves to be difficult; the problem has a quadratic number of variables (one for each pair of utterances) and a cubic number of triangle inequality constraints (three for each triplet). With 800 utterances in our test set, even solving the linear program with CPLEX (Ilog, Inc., 2003) is too expensive to be practical.

Although there are a variety of approximations and local searches, we do not wish to investigate partitioning methods in this paper, so we simply use a greedy search. In this algorithm, we assign utterance  $j$  by examining all previous utterances  $i$  within the classifier’s window, and treating the classifier’s judgement  $p_{i,j} - .5$  as a vote for  $cluster(i)$ . If the maximum vote is greater than 0, we set  $cluster(j) = argmax_c vote_c$ . Otherwise  $j$  is put in a new cluster. Greedy clustering makes at least a reasonable starting point for further efforts, since it is a natural online algorithm— it assigns each utterance as it arrives, without reference to the future.

At any rate, we should not take our objective function too seriously. Although it is roughly correlated with performance, the high error rate of the classifier makes it unlikely that small changes in objective will mean much. In fact, the objective value of our output solutions are generally higher than those for true so-

<sup>8</sup>We set up the problem by taking the weight of edge  $i, j$  as the classifier’s decision  $p_{i,j} - .5$ . Roth and Yih (2004) use log probabilities as weights. Bansal et al. (2004) propose the log odds ratio  $log(p/(1 - p))$ . We are unsure of the relative merit of these approaches.

lutions, which implies we have already reached the limits of what our classifier can tell us.

## 5 Experiments

We annotate the 800 line test transcript using our system. The annotation obtained has 63 conversations, with mean length 12.70. The average density of conversations is 2.9, and the entropy is 3.79. This places it within the bounds of our human annotations (see table 1), toward the more general end of the spectrum.

As a standard of comparison for our system, we provide results for several baselines— trivial systems which any useful annotation should outperform.

**All different** Each utterance is a separate conversation.

**All same** The whole transcript is a single conversation.

**Blocks of  $k$**  Each consecutive group of  $k$  utterances is a conversation.

**Pause of  $k$**  Each pause of  $k$  seconds or more separates two conversations.

**Speaker** Each speaker’s utterances are treated as a monologue.

For each particular metric, we calculate the best baseline result among all of these. To find the best block size or pause length, we search over multiples of 5 between 5 and 300. This makes these baselines appear better than they really are, since their performance is optimized with respect to the test data.

Our results, in table 3, are encouraging. On average, annotators agree more with each other than with any artificial annotation, and more with our model than with the baselines. For the 1-to-1 accuracy metric, we cannot claim much beyond these general results. The range of human variation is quite wide, and there are annotators who are closer to baselines than to any other human annotator. As explained earlier, this is because some human annotations are much more specific than others. For very specific annotations, the best baselines are short blocks or pauses. For the most general, marking all utterances the same does very well (although for all other annotations, it is extremely poor).

	Other Annotators	Model	Best Baseline	All Diff	All Same
Mean 1-to-1	52.98	40.62	34.73 (Blocks of 40)	10.16	20.93
Max 1-to-1	63.50	51.12	56.00 (Pause of 65)	16.00	53.50
Min 1-to-1	35.63	33.63	28.62 (Pause of 25)	6.25	7.13
Mean $loc_3$	81.09	72.75	62.16 (Speaker)	52.93	47.07
Max $loc_3$	86.53	75.16	69.05 (Speaker)	62.15	57.47
Min $loc_3$	74.75	70.47	54.37 (Speaker)	42.53	37.85

Table 3: Metric values between proposed annotations and human annotations. Model scores typically fall between inter-annotator agreement and baseline performance.

For the local metric, the results are much clearer. There is no overlap in the ranges; for every test annotation, agreement is highest with other annotator, then our model and finally the baselines. The most competitive baseline is one conversation per speaker, which makes sense, since if a speaker makes two comments in a four-utterance window, they are very likely to be related.

The name mention features are critical for our model’s performance. Without this feature, the classifier’s development F-score drops from 71 to 56. The disentanglement system’s test performance decreases proportionally; mean 1-to-1 falls to 36.08, and mean  $loc_3$  to 63.00, essentially baseline performance. On the other hand, mentions are not sufficient; with only name mention and time gap features, mean 1-to-1 is 38.54 and  $loc_3$  is 67.14. For some utterances, of course, name mentions provide the only reasonable clue to the correct decision, which is why humans mention names in the first place. But our system is probably overly dependent on them, since they are very reliable compared to our other features.

## 6 Future Work

Although our annotators are reasonably reliable, it seems clear that they think of conversations as a hierarchy, with digressions and schisms. We are interested to see an annotation protocol which more closely follows human intuition and explicitly includes these kinds of relationships.

We are also interested to see how well this feature set performs on speech data, as in (Aoki et al., 2003). Spoken conversation is more natural than text chat, but when participants are not face-to-face, disentanglement remains a problem. On the other hand, spoken dialogue contains new sources of information,

such as prosody. Turn-taking behavior is also more distinct, which makes the task easier, but according to (Aoki et al., 2006), it is certainly not sufficient.

Improving the current model will definitely require better features for the classifier. However, we also left the issue of partitioning nearly completely unexplored. If the classifier can indeed be improved, we expect the impact of search errors to increase. Another issue is that human users may prefer more or less specific annotations than our model provides. We have observed that we can produce lower or higher-entropy annotations by changing the classifier’s bias to label more edges same or different. But we do not yet know whether this corresponds with human judgements, or merely introduces errors.

## 7 Conclusion

This work provides a corpus of annotated data for chat disentanglement, which, along with our proposed metrics, should allow future researchers to evaluate and compare their results quantitatively<sup>9</sup>. Our annotations are consistent with one another, especially with respect to local agreement. We show that features based on discourse patterns and the content of utterances are helpful in disentanglement. The model we present can outperform a variety of baselines.

## Acknowledgements

Our thanks to Suman Karumuri, Steve Sloman, Matt Lease, David McClosky, 7 test annotators, 3 pilot annotators, 3 anonymous reviewers and the NSF PIRE grant.

<sup>9</sup>Code and data for this project will be available at <http://cs.brown.edu/people/melsner>.

## References

- Evrin Acar, Seyit Ahmet Camtepe, Mukkai S. Krishnamoorthy, and Blent Yener. 2005. Modeling and multiway analysis of chatroom tensors. In Paul B. Kantor, Gheorghe Muresan, Fred Roberts, Daniel Dajun Zeng, Fei-Yue Wang, Hsinchun Chen, and Ralph C. Merkle, editors, *ISI*, volume 3495 of *Lecture Notes in Computer Science*, pages 256–268. Springer.
- Paul M. Aoki, Matthew Romaine, Margaret H. Szymanski, James D. Thornton, Daniel Wilson, and Allison Woodruff. 2003. The mad hatter’s cocktail party: a social mobile audio space supporting multiple simultaneous conversations. In *CHI ’03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 425–432, New York, NY, USA. ACM Press.
- Paul M. Aoki, Margaret H. Szymanski, Luke D. Plurkowski, James D. Thornton, Allison Woodruff, and Weilie Yi. 2006. Where’s the “party” in “multi-party”? analyzing the structure of small-group social talk. In *CSCW ’06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 393–402, New York, NY, USA. ACM Press.
- Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Machine Learning*, 56(1-3):89–113.
- Seyit Ahmet Camtepe, Mark K. Goldberg, Malik Magdon-Ismael, and Mukkai Krishnamoorthy. 2005. Detecting conversing groups of chatters: a model, algorithms, and tests. In *IADIS AC*, pages 89–96.
- 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/>, August.
- Peter Foltz, Walter Kintsch, and Thomas Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2&3):285–307.
- Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *ACL ’03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 562–569, Morristown, NJ, USA. Association for Computational Linguistics.
- David Graff. 1995. *North American News Text Corpus*. Linguistic Data Consortium. LDC95T21.
- Ilog, Inc. 2003. Cplex solver.
- Natasa Jovanovic, Rieks op den Akker, and Anton Nijholt. 2006. Addressee identification in face-to-face meetings. In *EACL*. The Association for Computer Linguistics.
- Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *ACL*. The Association for Computer Linguistics.
- Jacki O’Neill and David Martin. 2003. Text chat in action. In *GROUP ’03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, pages 40–49, New York, NY, USA. ACM Press.
- Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL-2004*, pages 1–8. Boston, MA, USA.
- Harvey Sacks, Emanuel A. Schegloff, and Gail Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735.

# An Entity-Mention Model for Coreference Resolution with Inductive Logic Programming

Xiaofeng Yang<sup>1</sup> Jian Su<sup>1</sup> Jun Lang<sup>2</sup>  
Chew Lim Tan<sup>3</sup> Ting Liu<sup>2</sup> Sheng Li<sup>2</sup>

<sup>1</sup>Institute for Infocomm Research  
{xiaofeng,yujian}@i2r.a-star.edu.sg

<sup>2</sup>Harbin Institute of Technology  
{bill\_lang,tliu}@ir.hit.edu.cn  
lisheng@hit.edu.cn

<sup>3</sup>National University of Singapore,  
tancl@comp.nus.edu.sg

## Abstract

The traditional mention-pair model for coreference resolution cannot capture information beyond mention pairs for both learning and testing. To deal with this problem, we present an expressive entity-mention model that performs coreference resolution at an entity level. The model adopts the Inductive Logic Programming (ILP) algorithm, which provides a relational way to organize different knowledge of entities and mentions. The solution can explicitly express relations between an entity and the contained mentions, and automatically learn first-order rules important for coreference decision. The evaluation on the ACE data set shows that the ILP based entity-mention model is effective for the coreference resolution task.

## 1 Introduction

Coreference resolution is the process of linking multiple mentions that refer to the same entity. Most of previous work adopts the mention-pair model, which recasts coreference resolution to a binary classification problem of determining whether or not two mentions in a document are co-referring (e.g. Aone and Bennett (1995); McCarthy and Lehnert (1995); Soon et al. (2001); Ng and Cardie (2002)). Although having achieved reasonable success, the mention-pair model has a limitation that information beyond mention pairs is ignored for training and testing. As an individual mention usually lacks adequate descriptive information of the referred entity, it is often difficult to judge whether or not two men-

tions are talking about the same entity simply from the pair alone.

An alternative learning model that can overcome this problem performs coreference resolution based on entity-mention pairs (Luo et al., 2004; Yang et al., 2004b). Compared with the traditional mention-pair counterpart, the entity-mention model aims to make coreference decision at an entity level. Classification is done to determine whether a mention is a referent of a partially found entity. A mention to be resolved (called *active mention* henceforth) is linked to an appropriate entity chain (if any), based on classification results.

One problem that arises with the entity-mention model is how to represent the knowledge related to an entity. In a document, an entity may have more than one mention. It is impractical to enumerate all the mentions in an entity and record their information in a single feature vector, as it would make the feature space too large. Even worse, the number of mentions in an entity is not fixed, which would result in variant-length feature vectors and make trouble for normal machine learning algorithms. A solution seen in previous work (Luo et al., 2004; Culotta et al., 2007) is to design a set of first-order features summarizing the information of the mentions in an entity, for example, “whether the entity has any mention that is a name alias of the active mention?” or “whether most of the mentions in the entity have the same head word as the active mention?” These features, nevertheless, are designed in an ad-hoc manner and lack the capability of describing each individual mention in an entity.

In this paper, we present a more expressive entity-

mention model for coreference resolution. The model employs Inductive Logic Programming (ILP) to represent the relational knowledge of an active mention, an entity, and the mentions in the entity. On top of this, a set of first-order rules is automatically learned, which can capture the information of each individual mention in an entity, as well as the global information of the entity, to make coreference decision. Hence, our model has a more powerful representation capability than the traditional mention-pair or entity-mention model. And our experimental results on the ACE data set shows the model is effective for coreference resolution.

## 2 Related Work

There are plenty of learning-based coreference resolution systems that employ the mention-pair model. A typical one of them is presented by Soon et al. (2001). In the system, a training or testing instance is formed for two mentions in question, with a feature vector describing their properties and relationships. At a testing time, an active mention is checked against all its preceding mentions, and is linked with the closest one that is classified as positive. The work is further enhanced by Ng and Cardie (2002) by expanding the feature set and adopting a “best-first” linking strategy.

Recent years have seen some work on the entity-mention model. Luo et al. (2004) propose a system that performs coreference resolution by doing search in a large space of entities. They train a classifier that can determine the likelihood that an active mention should belong to an entity. The entity-level features are calculated with an “Any-X” strategy: an entity-mention pair would be assigned a feature X, if any mention in the entity has the feature X with the active mention.

Culotta et al. (2007) present a system which uses an online learning approach to train a classifier to judge whether two entities are coreferential or not. The features describing the relationships between two entities are obtained based on the information of every possible pair of mentions from the two entities. Different from (Luo et al., 2004), the entity-level features are computed using a “Most-X” strategy, that is, two given entities would have a feature X, if most of the mention pairs from the two entities

have the feature X.

Yang et al. (2004b) suggest an entity-based coreference resolution system. The model adopted in the system is similar to the mention-pair model, except that the entity information (e.g., the global number/gender agreement) is considered as additional features of a mention in the entity.

McCallum and Wellner (2003) propose several graphical models for coreference analysis. These models aim to overcome the limitation that pairwise coreference decisions are made independently of each other. The simplest model conditions coreference on mention pairs, but enforces dependency by calculating the distance of a node to a partition (i.e., the probability that an active mention belongs to an entity) based on the sum of its distances to all the nodes in the partition (i.e., the sum of the probability of the active mention co-referring with the mentions in the entity).

Inductive Logic Programming (ILP) has been applied to some natural language processing tasks, including parsing (Mooney, 1997), POS disambiguation (Cussens, 1996), lexicon construction (Claveau et al., 2003), WSD (Specia et al., 2007), and so on. However, to our knowledge, our work is the first effort to adopt this technique for the coreference resolution task.

## 3 Modelling Coreference Resolution

Suppose we have a document containing  $n$  mentions  $\{m_j : 1 < j < n\}$ , in which  $m_j$  is the  $j$ th mention occurring in the document. Let  $e_i$  be the  $i$ th entity in the document. We define

$$P(L|e_i, m_j), \quad (1)$$

the probability that a mention belongs to an entity. Here the random variable  $L$  takes a binary value and is 1 if  $m_j$  is a mention of  $e_i$ .

By assuming that mentions occurring after  $m_j$  have no influence on the decision of linking  $m_j$  to an entity, we can approximate (1) as:

$$\begin{aligned} & P(L|e_i, m_j) \\ \propto & P(L|\{m_k \in e_i, 1 \leq k \leq j-1\}, m_j) \quad (2) \\ \propto & \max_{m_k \in e_i, 1 \leq k \leq j-1} P(L|m_k, m_j) \quad (3) \end{aligned}$$

(3) further assumes that an entity-mention score can be computed by using the maximum mention-



---

[ *Microsoft Corp.* ]<sub>1</sub><sup>1</sup> announced [ [ *its* ]<sub>2</sub><sup>1</sup> *new CEO* ]<sub>3</sub><sup>2</sup>  
 [ *yesterday* ]<sub>4</sub><sup>3</sup>. [ *The company* ]<sub>5</sub><sup>1</sup> said [ *he* ]<sub>6</sub><sup>2</sup> will ...

---

Table 1: A sample text

pair score. Both (2) and (1) can be approximated with a machine learning method, leading to the traditional mention-pair model and the entity-mention model for coreference resolution, respectively.

The two models will be described in the next subsections, with the sample text in Table 1 used for demonstration. In the table, a mention  $m$  is highlighted as  $[ m ]_{mid}^{eid}$ , where  $mid$  and  $eid$  are the IDs for the mention and the entity to which it belongs, respectively. Three entity chains can be found in the text, that is,

- e1** : *Microsoft Corp.* - *its* - *The company*
- e2** : *its new CEO* - *he*
- e3** : *yesterday*

### 3.1 Mention-Pair Model

As a baseline, we first describe a learning framework with the mention-pair model as adopted in the work by Soon et al. (2001) and Ng and Cardie (2002).

In the learning framework, a training or testing instance has the form of  $i\{m_k, m_j\}$ , in which  $m_j$  is an active mention and  $m_k$  is a preceding mention. An instance is associated with a vector of features, which is used to describe the properties of the two mentions as well as their relationships. Table 2 summarizes the features used in our study.

For training, given each encountered anaphoric mention  $m_j$  in a document, one single positive training instance is created for  $m_j$  and its closest antecedent. And a group of negative training instances is created for every intervening mentions between  $m_j$  and the antecedent. Consider the example text in Table 1, for the pronoun “*he*”, three instances are generated:  $i(\text{“The company”}, \text{“he”})$ ,  $i(\text{“yesterday”}, \text{“he”})$ , and  $i(\text{“its new CEO”}, \text{“he”})$ . Among them, the first two are labelled as negative while the last one is labelled as positive.

Based on the training instances, a binary classifier can be generated using any discriminative learning algorithm. During resolution, an input document is processed from the first mention to the last. For each

encountered mention  $m_j$ , a test instance is formed for each preceding mention,  $m_k$ . This instance is presented to the classifier to determine the coreference relationship.  $m_j$  is linked with the mention that is classified as positive (if any) with the highest confidence value.

### 3.2 Entity-Mention Model

The mention-based solution has a limitation that information beyond a mention pair cannot be captured. As an individual mention usually lacks complete description about the referred entity, the coreference relationship between two mentions may be not clear, which would affect classifier learning. Consider a document with three coreferential mentions “*Mr. Powell*”, “*he*”, and “*Powell*”, appearing in that order. The positive training instance  $i(\text{“he”}, \text{“Powell”})$  is not informative, as the pronoun “*he*” itself discloses nothing but the gender. However, if the whole entity is considered instead of only one mention, we can know that “*he*” refers to a male person named “*Powell*”. And consequently, the coreference relationships between the mentions would become more obvious.

The mention-pair model would also cause errors at a testing time. Suppose we have three mentions “*Mr. Powell*”, “*Powell*”, and “*she*” in a document. The model tends to link “*she*” with “*Powell*” because of their proximity. This error can be avoided, if we know “*Powell*” belongs to the entity starting with “*Mr. Powell*”, and therefore refers to a male person and cannot co-refer with “*she*”.

The entity-mention model based on Eq. (2) performs coreference resolution at an entity-level. For simplicity, the framework considered for the entity-mention model adopts similar training and testing procedures as for the mention-pair model. Specifically, a training or testing instance has the form of  $i\{e_i, m_j\}$ , in which  $m_j$  is an active mention and  $e_i$  is a partial entity found before  $m_j$ . During training, given each anaphoric mention  $m_j$ , one single positive training instance is created for the entity to which  $m_j$  belongs. And a group of negative training instances is created for every partial entity whose last mention occurs between  $m_j$  and the closest antecedent of  $m_j$ .

See the sample in Table 1 again. For the pronoun “*he*”, the following three instances are generated for

Features describing an active mention, $m_j$	
defNP <sub>_mj</sub>	1 if $m_j$ is a definite description; else 0
indefNP <sub>_mj</sub>	1 if $m_j$ is an indefinite NP; else 0
nameNP <sub>_mj</sub>	1 if $m_j$ is a named-entity; else 0
pron <sub>_mj</sub>	1 if $m_j$ is a pronoun; else 0
bareNP <sub>_mj</sub>	1 if $m_j$ is a bare NP (i.e., NP without determiners); else 0
Features describing a previous mention, $m_k$	
defNP <sub>_mk</sub>	1 if $m_k$ is a definite description; else 0
indefNP <sub>_mk</sub>	1 if $m_k$ is an indefinite NP; else 0
nameNP <sub>_mk</sub>	1 if $m_k$ is a named-entity; else 0
pron <sub>_mk</sub>	1 if $m_k$ is a pronoun; else 0
bareNP <sub>_mk</sub>	1 if $m_k$ is a bare NP; else 0
subject <sub>_mk</sub>	1 if $m_k$ is an NP in a subject position; else 0
Features describing the relationships between $m_k$ and $m_j$	
sentDist	sentence distance between two mentions
numAgree	1 if two mentions match in the number agreement; else 0
genderAgree	1 if two mentions match in the gender agreement; else 0
parallelStruct	1 if two mentions have an identical collocation pattern; else 0
semAgree	1 if two mentions have the same semantic category; else 0
nameAlias	1 if two mentions are an alias of the other; else 0
apposition	1 if two mentions are in an appositive structure; else 0
predicative	1 if two mentions are in a predicative structure; else 0
strMatch_Head	1 if two mentions have the same head string; else 0
strMatch_Full	1 if two mentions contain the same strings, excluding the determiners; else 0
strMatch_Contain	1 if the string of $m_j$ is fully contained in that of $m_k$ ; else 0

Table 2: Feature set for coreference resolution

entity  $e1$ ,  $e3$  and  $e2$ :

$i(\{\text{"Microsoft Corp."}, \text{"its"}, \text{"The company"}\}, \text{"he"}),$   
 $i(\{\text{"yesterday"}\}, \text{"he"}),$   
 $i(\{\text{"its new CEO"}\}, \text{"he"}).$

Among them, the first two are labelled as negative, while the last one is positive.

The resolution is done using a greedy clustering strategy. Given a test document, the mentions are processed one by one. For each encountered mention  $m_j$ , a test instance is formed for each partial entity found so far,  $e_i$ . This instance is presented to the classifier.  $m_j$  is appended to the entity that is classified as positive (if any) with the highest confidence value. If no positive entity exists, the active mention is deemed as non-anaphoric and forms a new entity. The process continues until the last mention of the document is reached.

One potential problem with the entity-mention model is how to represent the entity-level knowledge. As an entity may contain more than one candidate and the number is not fixed, it is impractical to enumerate all the mentions in an entity and put their properties into a single feature vector. As a baseline, we follow the solution proposed in (Luo et al., 2004) to design a set of first-order features. The features are similar to those for the mention-pair model as shown in Table 2, but their values are calculated at an entity level. Specifically, the lexical and grammatical features are computed by testing any mention<sup>1</sup> in the entity against the active mention, for ex-

ample, the feature *nameAlias* is assigned value 1 if at least one mention in the entity is a name alias of the active mention. The distance feature (i.e., *sentDist*) is the minimum distance between the mentions in the entity and the active mention.

The above entity-level features are designed in an ad-hoc way. They cannot capture the detailed information of each individual mention in an entity. In the next section, we will present a more expressive entity-mention model by using ILP.

## 4 Entity-mention Model with ILP

### 4.1 Motivation

The entity-mention model based on Eq. (2) requires relational knowledge that involves information of an active mention ( $m_j$ ), an entity ( $e_i$ ), and the mentions in the entity ( $\{m_k \in e_i\}$ ). However, normal machine learning algorithms work on attribute-value vectors, which only allows the representation of atomic proposition. To learn from relational knowledge, we need an algorithm that can express first-order logic. This requirement motivates our use of Inductive Logic Programming (ILP), a learning algorithm capable of inferring logic programs. The relational nature of ILP makes it possible to explicitly represent relations between an entity and its mentions, and thus provides a powerful expressiveness for the coreference resolution task.

reference relationship with antecedents in a local discourse. Hence, if an active mention is a pronoun, we only consider the mentions in its previous two sentences for feature computation.

<sup>1</sup>Linguistically, pronouns usually have the most direct coref-

ILP uses logic programming as a uniform representation for examples, background knowledge and hypotheses. Given a set of positive and negative example  $E = E^+ \cup E^-$ , and a set of background knowledge  $K$  of the domain, ILP tries to induce a set of hypotheses  $h$  that covers most of  $E^+$  with no  $E^-$ , i.e.,  $K \wedge h \models E^+$  and  $K \wedge h \not\models E^-$ .

In our study, we choose ALEPH<sup>2</sup>, an ILP implementation by Srinivasan (2000) that has been proven well suited to deal with a large amount of data in multiple domains. For its routine use, ALEPH follows a simple procedure to induce rules. It first selects an example and builds the most specific clause that entertains the example. Next, it tries to search for a clause more general than the bottom one. The best clause is added to the current theory and all the examples made redundant are removed. The procedure repeats until all examples are processed.

## 4.2 Apply ILP to coreference resolution

Given a document, we encode a mention or a partial entity with a unique constant. Specifically,  $m_j$  represents the  $j$ th mention (e.g.,  $m_6$  for the pronoun “he”).  $e_{i-j}$  represents the partial entity  $i$  before the  $j$ th mention. For example,  $e_{1-6}$  denotes the part of  $e_1$  before  $m_6$ , i.e., {“Microsoft Corp.”, “its”, “the company”}, while  $e_{1-5}$  denotes the part of  $e_1$  before  $m_5$  (“The company”), i.e., {“Microsoft Corp.”, “its”}.

Training instances are created as described in Section 3.2 for the entity-mention model. Each instance is recorded with a predicate  $link(e_{i-j}, m_j)$ , where  $m_j$  is an active mention and  $e_{i-j}$  is a partial entity. For example, the three training instances formed by the pronoun “he” are represented as follows:

$link(e_{1-6}, m_6)$ .

$link(e_{3-6}, m_6)$ .

$link(e_{2-6}, m_6)$ .

The first two predicates are put into  $E^-$ , while the last one is put to  $E^+$ .

The background knowledge for an instance  $link(e_{i-j}, m_j)$  is also represented with predicates, which are divided into the following types:

1. Predicates describing the information related to  $e_{i-j}$  and  $m_j$ . The properties of  $m_j$  are pre-

<sup>2</sup>[http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph\\_toc.html](http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph_toc.html)

sented with predicates like  $f(m, v)$ , where  $f$  corresponds to a feature in the first part of Table 2 (removing the suffix  $_mj$ ), and  $v$  is its value. For example, the pronoun “he” can be described by the following predicates:

$defNP(m_6, 0)$ .  $indefNP(m_6, 0)$ .

$nameNP(m_6, 0)$ .  $pron(m_6, 1)$ .

$bareNP(m_6, 0)$ .

The predicates for the relationships between  $e_{i-j}$  and  $m_j$  take a form of  $f(e, m, v)$ . In our study, we consider the number agreement (*entNumAgree*) and the gender agreement (*entGenderAgree*) between  $e_{i-j}$  and  $m_j$ .  $v$  is 1 if all of the mentions in  $e_{i-j}$  have consistent number/gender agreement with  $m_j$ , e.g.,  $entNumAgree(e_{1-6}, m_6, 1)$ .

2. Predicates describing the belonging relations between  $e_{i-j}$  and its mentions. A predicate  $has\_mention(e, m)$  is used for each mention in  $e$ <sup>3</sup>. For example, the partial entity  $e_{1-6}$  has three mentions,  $m_1$ ,  $m_2$  and  $m_5$ , which can be described as follows:

$has\_mention(e_{1-6}, m_1)$ .

$has\_mention(e_{1-6}, m_2)$ .

$has\_mention(e_{1-6}, m_5)$ .

3. Predicates describing the information related to  $m_j$  and each mention  $m_k$  in  $e_{i-j}$ . The predicates for the properties of  $m_k$  correspond to the features in the second part of Table 2 (removing the suffix  $_mk$ ), while the predicates for the relationships between  $m_j$  and  $m_k$  correspond to the features in the third part of Table 2. For example, given the two mentions  $m_1$  (“Microsoft Corp.”) and  $m_6$  (“he”), the following predicates can be applied:

$nameNP(m_1, 1)$ .

$pron(m_1, 0)$ .

...

$nameAlias(m_1, m_6, 0)$ .

$sentDist(m_1, m_6, 1)$ .

...

the last two predicates represent that  $m_1$  and

<sup>3</sup>If an active mention  $m_j$  is a pronoun, only the previous mentions in two sentences apart are recorded by *has\\_mention*, while the farther ones are ignored as they have less impact on the resolution of the pronoun.

$m_6$  are not name alias, and are one sentence apart.

By using the three types of predicates, the different knowledge related to entities and mentions are integrated. The predicate *has\_mention* acts as a bridge connecting the entity-mention knowledge and the mention-pair knowledge. As a result, when evaluating the coreference relationship between an active mention and an entity, we can make use of the “global” information about the entity, as well as the “local” information of each individual mention in the entity.

From the training instances and the associated background knowledge, a set of hypotheses can be automatically learned by ILP. Each hypothesis is output as a rule that may look like:

*link(A,B):-*  
*predi1, predi2, ..., has\_mention(A,C), ..., prediN.*  
 which corresponds to first-order logic  
 $\forall A, B (predi1 \wedge predi2 \wedge \dots \wedge$   
 $\exists C (has\_mention(A, C) \wedge \dots \wedge prediN)$   
 $\rightarrow link(A, B))$

Consider an example rule produced in our system:

*link(A,B) :-*  
*has\_mention(A,C), numAgree(B,C,1),*  
*strMatch\_Head(B,C,1), bareNP(C,1).*

Here, variables  $A$  and  $B$  stand for an entity and an active mention in question. The first-order logic is implemented by using non-instantiated arguments  $C$  in the predicate *has\_mention*. This rule states that a mention  $B$  should belong to an entity  $A$ , if there exists a mention  $C$  in  $A$  such that  $C$  is a bare noun phrase with the same head string as  $B$ , and matches in number with  $B$ . In this way, the detailed information of each individual mention in an entity can be captured for resolution.

A rule is applicable to an instance  $link(e, m)$ , if the background knowledge for the instance can be described by the predicates in the body of the rule. Each rule is associated with a score, which is the accuracy that the rule can produce for the training instances.

The learned rules are applied to resolution in a similar way as described in Section 3.2. Given an active mention  $m$  and a partial entity  $e$ , a test instance  $link(e, m)$  is formed and tested against every rule in the rule set. The confidence that  $m$  should

	Train		Test	
	#entity	#mention	#entity	#mention
NWire	1678	9861	411	2304
NPaper	1528	10277	365	2290
BNews	1695	8986	468	2493

Table 3: statistics of entities (length > 1) and contained mentions

belong to  $e$  is the maximal score of the applicable rules. An active mention is linked to the entity with the highest confidence value (above 0.5), if any.

## 5 Experiments and Results

### 5.1 Experimental Setup

In our study, we did evaluation on the ACE-2003 corpus, which contains two data sets, training and devtest, used for training and testing respectively. Each of these sets is further divided into three domains: newswire (NWire), newspaper (NPaper), and broadcast news (BNews). The number of entities with more than one mention, as well as the number of the contained mentions, is summarized in Table 3.

For both training and resolution, an input raw document was processed by a pipeline of NLP modules including Tokenizer, Part-of-Speech tagger, NP Chunker and Named-Entity (NE) Recognizer. Trained and tested on Penn WSJ TreeBank, the POS tagger could obtain an accuracy of 97% and the NP chunker could produce an F-measure above 94% (Zhou and Su, 2000). Evaluated for the MUC-6 and MUC-7 Named-Entity task, the NER module (Zhou and Su, 2002) could provide an F-measure of 96.6% (MUC-6) and 94.1% (MUC-7). For evaluation, Vilain et al. (1995)’s scoring algorithm was adopted to compute recall and precision rates.

By default, the ALEPH algorithm only generates rules that have 100% accuracy for the training data. And each rule contains at most three predicates. To accommodate for coreference resolution, we loosened the restrictions to allow rules that have above 50% accuracy and contain up to ten predicates. Default parameters were applied for all the other settings in ALEPH as well as other learning algorithms used in the experiments.

### 5.2 Results and Discussions

Table 4 lists the performance of different coreference resolution systems. For comparison, we first

	NWire			NPaper			BNews		
	R	P	F	R	P	F	R	P	F
C4.5									
- Mention-Pair	68.2	54.3	60.4	67.3	50.8	57.9	66.5	59.5	62.9
- Entity-Mention	66.8	55.0	60.3	64.2	53.4	58.3	64.6	60.6	62.5
- Mention-Pair (all mentions in entity)	66.7	49.3	56.7	65.8	48.9	56.1	66.5	47.6	55.4
ILP									
- Mention-Pair	66.1	54.8	59.5	65.6	54.8	59.7	63.5	60.8	62.1
- Entity-Mention	65.0	58.9	<b>61.8</b>	63.4	57.1	<b>60.1</b>	61.7	65.4	<b>63.5</b>

Table 4: Results of different systems for coreference resolution

examined the C4.5 algorithm<sup>4</sup> which is widely used for the coreference resolution task. The first line of the table shows the baseline system that employs the traditional mention-pair model (MP) as described in Section 3.1. From the table, our baseline system achieves a recall of around 66%-68% and a precision of around 50%-60%. The overall F-measure for NWire, NPaper and BNews is 60.4%, 57.9% and 62.9% respectively. The results are comparable to those reported in (Ng, 2005) which uses similar features and gets an F-measure ranging in 50-60% for the same data set. As our system relies only on simple and knowledge-poor features, the achieved F-measure is around 2-4% lower than the state-of-the-art systems do, like (Ng, 2007) and (Yang and Su, 2007) which utilized sophisticated semantic or real-world knowledge. Since ILP has a strong capability in knowledge management, our system could be further improved if such helpful knowledge is incorporated, which will be explored in our future work.

The second line of Table 4 is for the system that employs the entity-mention model (EM) with “Any-X” based entity features, as described in Section 3.2. We can find that the EM model does not show superiority over the baseline MP model. It achieves a higher precision (up to 2.6%), but a lower recall (2.9%), than MP. As a result, we only see  $\pm 0.4\%$  difference between the F-measure. The results are consistent with the reports by Luo et al. (2004) that the entity-mention model with the “Any-X” first-order features performs worse than the normal mention-pair model. In our study, we also tested the “Most-X” strategy for the first-order features as in (Culotta et al., 2007), but got similar results without much difference ( $\pm 0.5\%$  F-measure) in perfor-

mance. Besides, as with our entity-mention predicates described in Section 4.2, we also tried the “All-X” strategy for the entity-level agreement features, that is, whether all mentions in a partial entity agree in number and gender with an active mention. However, we found this bring no improvement against the “Any-X” strategy.

As described, given an active mention  $m_j$ , the MP model only considers the mentions between  $m_j$  and its closest antecedent. By contrast, the EM model considers not only these mentions, but also their antecedents in the same entity link. We were interested in examining what if the MP model utilizes all the mentions in an entity as the EM model does. As shown in the third line of Table 4, such a solution damages the performance; while the recall is at the same level, the precision drops significantly (up to 12%) and as a result, the F-measure is even lower than the original MP model. This should be because a mention does not necessarily have direct coreference relationships with all of its antecedents. As the MP model treats each mention-pair as an independent instance, including all the antecedents would produce many less-confident positive instances, and thus adversely affect training.

The second block of the table summarizes the performance of the systems with ILP. We were first concerned with how well ILP works for the mention-pair model, compared with the normally used algorithm C4.5. From the results shown in the fourth line of Table 4, ILP exhibits the same capability in the resolution; it tends to produce a slightly higher precision but a lower recall than C4.5 does. Overall, it performs better in F-measure (1.8%) for Npaper, while slightly worse ( $< 1\%$ ) for Nwire and BNews. These results demonstrate that ILP could be used as

<sup>4</sup><http://www.rulequest.com/see5-info.html>

```

link(A,B) :-
bareNP(B,0), has_mention(A,C), appositive(C,1).

link(A,B) :-
has_mention(A,C), numAgree(B,C,1), strMatch_Head(B,C,1), bareNP(C,1).

link(A,B) :-
nameNP(B,0), has_mention(A,C), predicative(C,1).

link(A,B) :-
has_mention(A,C), strMatch_Contain(B,C,1), strMatch_Head(B,C,1), bareNP(C,0).

link(A,B) :-
nameNP(B,0), has_mention(A,C), nameAlias(C,1), bareNP(C,0).

link(A,B) :-
pron(B,1), has_mention(A,C), nameNP(C,1), has_mention(A,D), indefNP(D,1),
subject(D, 1).
...

```

Figure 1: Examples of rules produced by ILP (entity-mention model)

a good classifier learner for the mention-pair model.

The fifth line of Table 4 is for the ILP based entity-mention model (described in Section 4.2). We can observe that the model leads to a better performance than all the other models. Compared with the system with the MP model (under ILP), the EM version is able to achieve a higher precision (up to 4.6% for BNews). Although the recall drops slightly (up to 1.8% for BNews), the gain in the precision could compensate it well; it beats the MP model in the overall F-measure for all three domains (2.3% for Nwire, 0.4% for Npaper, 1.4% for BNews). Especially, the improvement in NWire and BNews is statistically significant under a 2-tailed  $t$  test ( $p < 0.05$ ). Compared with the EM model with the manually designed first-order feature (the second line), the ILP-based EM solution also yields better performance in precision (with a slightly lower recall) as well as the overall F-measure (1.0% - 1.8%).

The improvement in precision against the mention-pair model confirms that the global information beyond a single mention pair, when being considered for training, can make coreference relations clearer and help classifier learning. The better performance against the EM model with heuristically designed features also suggests that ILP is able to learn effective first-order rules for the coreference resolution task.

In Figure 1, we illustrate part of the rules produced by ILP for the entity-mention model (NWire domain), which shows how the relational knowledge of entities and mentions is represented for decision making. An interesting finding, as shown in the last

rule of the table, is that multiple non-instantiated arguments (i.e.  $C$  and  $D$ ) could possibly appear in the same rule. According to this rule, a pronominal mention should be linked with a partial entity which contains a named-entity and contains an indefinite NP in a subject position. This supports the claims in (Yang et al., 2004a) that coreferential information is an important factor to evaluate a candidate antecedent in pronoun resolution. Such complex logic makes it possible to capture information of multiple mentions in an entity at the same time, which is difficult to implemented in the mention-pair model and the ordinary entity-mention model with heuristic first-order features.

## 6 Conclusions

This paper presented an expressive entity-mention model for coreference resolution by using Inductive Logic Programming. In contrast to the traditional mention-pair model, our model can capture information beyond single mention pairs for both training and testing. The relational nature of ILP enables our model to explicitly express the relations between an entity and its mentions, and to automatically learn the first-order rules effective for the coreference resolution task. The evaluation on ACE data set shows that the ILP based entity-model performs better than the mention-pair model (with up to 2.3% increase in F-measure), and also beats the entity-mention model with heuristically designed first-order features.

Our current work focuses on the learning model that calculates the probability of a mention belonging to an entity. For simplicity, we just use a greedy clustering strategy for resolution, that is, a mention is linked to the current best partial entity. In our future work, we would like to investigate more sophisticated clustering methods that would lead to global optimization, e.g., by keeping a large search space (Luo et al., 2004) or using integer programming (Denis and Baldridge, 2007).

**Acknowledgements** This research is supported by a Specific Targeted Research Project (STREP) of the European Union’s 6th Framework Programme within IST call 4, Bootstrapping Of Ontologies and Terminologies STRategic REsearch Project (BOOT-Strep).

## References

- C. Aone and S. W. Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 122–129.
- V. Claveau, P. Sebillot, C. Fabre, and P. Bouillon. 2003. Learning semantic lexicons from a part-of-speech and semantically tagged corpus using inductive logic programming. *Journal of Machine Learning Research*, 4:493–525.
- A. Culotta, M. Wick, and A. McCallum. 2007. First-order probabilistic models for coreference resolution. In *Proceedings of the Annual Meeting of the North America Chapter of the Association for Computational Linguistics (NAACL)*, pages 81–88.
- J. Cussens. 1996. Part-of-speech disambiguation using ilp. Technical report, Oxford University Computing Laboratory.
- P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of the Annual Meeting of the North America Chapter of the Association for Computational Linguistics (NAACL)*, pages 236–243.
- X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 135–142.
- A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of IJCAI-03 Workshop on Information Integration on the Web*, pages 79–86.
- J. McCarthy and W. Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the 14th International Conference on Artificial Intelligences (IJCAI)*, pages 1050–1055.
- R. Mooney. 1997. Inductive logic programming for natural language processing. In *Proceedings of the sixth International Inductive Logic Programming Workshop*, pages 3–24.
- V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 104–111, Philadelphia.
- V. Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 157–164.
- V. Ng. 2007. Semantic class induction and coreference resolution. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 536–543.
- W. Soon, H. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.
- L. Specia, M. Stevenson, and M. V. Nunes. 2007. Learning expressive models for words sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 41–48.
- A. Srinivasan. 2000. The aleph manual. Technical report, Oxford University Computing Laboratory.
- M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 45–52, San Francisco, CA. Morgan Kaufmann Publishers.
- X. Yang and J. Su. 2007. Coreference resolution using semantic relatedness information from automatically discovered patterns. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 528–535.
- X. Yang, J. Su, G. Zhou, and C. Tan. 2004a. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 127–134, Barcelona.
- X. Yang, J. Su, G. Zhou, and C. Tan. 2004b. An NP-cluster approach to coreference resolution. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 219–225, Geneva.
- G. Zhou and J. Su. 2000. Error-driven HMM-based chunk tagger with context-dependent lexicon. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 71–79, Hong Kong.
- G. Zhou and J. Su. 2002. Named Entity recognition using a HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 473–480, Philadelphia.

# Gestural Cohesion for Topic Segmentation

Jacob Eisenstein, Regina Barzilay and Randall Davis

Computer Science and Artificial Intelligence Laboratory

Massachusetts Institute of Technology

77 Massachusetts Ave., Cambridge MA 02139

{jacobe, regina, davis}@csail.mit.edu

## Abstract

This paper explores the relationship between discourse segmentation and coverbal gesture. Introducing the idea of *gestural cohesion*, we show that coherent topic segments are characterized by homogeneous gestural forms and that changes in the distribution of gestural features predict segment boundaries. Gestural features are extracted automatically from video, and are combined with lexical features in a Bayesian generative model. The resulting multimodal system outperforms text-only segmentation on both manual and automatically-recognized speech transcripts.

## 1 Introduction

When people communicate face-to-face, discourse cues are expressed simultaneously through multiple channels. Previous research has extensively studied how discourse cues correlate with lexico-syntactic and prosodic features (Hearst, 1994; Hirschberg and Nakatani, 1998; Passonneau and Litman, 1997); this work informs various text and speech processing applications, such as automatic summarization and segmentation. Gesture is another communicative modality that frequently accompanies speech, yet it has not been exploited for computational discourse analysis.

This paper empirically demonstrates that gesture correlates with discourse structure. In particular, we show that automatically-extracted visual features can be combined with lexical cues in a statistical model to predict topic segmentation, a frequently studied form of discourse structure. Our

method builds on the idea that coherent discourse segments are characterized by *gestural cohesion*; in other words, that such segments exhibit homogeneous gestural patterns. Lexical cohesion (Halliday and Hasan, 1976) forms the backbone of many verbal segmentation algorithms, on the theory that segmentation boundaries should be placed where the distribution of words changes (Hearst, 1994). With gestural cohesion, we explore whether the same idea holds for gesture features.

The motivation for this approach comes from a series of psycholinguistic studies suggesting that gesture supplements speech with meaningful and unique semantic content (McNeill, 1992; Kendon, 2004). We assume that repeated patterns in gesture are indicative of the semantic coherence that characterizes well-defined discourse segments. An advantage of this view is that gestures can be brought to bear on discourse analysis without undertaking the daunting task of recognizing and interpreting individual gestures. This is crucial because coverbal gesture – unlike formal sign language – rarely follows any predefined form or grammar, and may vary dramatically by speaker.

A key implementational challenge is automatically extracting gestural information from raw video and representing it in a way that can be applied to discourse analysis. We employ a representation of *visual codewords*, which capture clusters of low-level motion patterns. For example, one codeword may correspond to strong left-right motion in the upper part of the frame. These codewords are then treated similarly to lexical items; our model identifies changes in their distribution, and predicts topic



boundaries appropriately. The overall framework is implemented as a hierarchical Bayesian model, supporting flexible integration of multiple knowledge sources.

Experimental results support the hypothesis that gestural cohesion is indicative of discourse structure. Applying our algorithm to a dataset of face-to-face dialogues, we find that gesture communicates unique information, improving segmentation performance over lexical features alone. The positive impact of gesture is most pronounced when automatically-recognized speech transcripts are used, but gestures improve performance by a significant margin even in combination with manual transcripts.

## 2 Related Work

**Gesture and discourse** Much of the work on gesture in natural language processing has focused on multimodal dialogue systems in which the gestures and speech may be constrained, e.g. (Johnston, 1998). In contrast, we focus on improving discourse processing on unconstrained natural language between humans. This effort follows basic psychological and linguistic research on the communicative role of gesture (McNeill, 1992; Kendon, 2004), including some efforts that made use of automatically acquired visual features (Quek, 2003). We extend these empirical studies with a statistical model of the relationship between gesture and discourse segmentation.

Hand-coded descriptions of body posture shifts and eye gaze behavior have been shown to correlate with topic and turn boundaries in task-oriented dialogue (Cassell et al., 2001). These findings are exploited to generate realistic conversational “grounding” behavior in an animated agent. The semantic content of gesture was leveraged – again, for gesture generation – in (Kopp et al., 2007), which presents an animated agent that is capable of augmenting navigation directions with gestures that describe the physical properties of landmarks along the route. Both systems generate plausible and human-like gestural behavior; we address the converse problem of *interpreting* such gestures.

In this vein, hand-coded gesture features have been used to improve sentence segmentation, show-

ing that sentence boundaries are unlikely to overlap gestures that are in progress (Chen et al., 2006). Features that capture the start and end of gestures are shown to improve sentence segmentation beyond lexical and prosodic features alone. This idea of gestural features as a sort of visual punctuation has parallels in the literature on prosody, which we discuss in the next subsection.

Finally, ambiguous noun phrases can be resolved by examining the similarity of co-articulated gestures (Eisenstein and Davis, 2007). While noun phrase coreference can be viewed as a discourse processing task, we address the higher-level discourse phenomenon of topic segmentation. In addition, this prior work focused primarily on pointing gestures directed at pre-printed visual aids. The current paper presents a new domain, in which speakers do not have access to visual aids. Thus pointing gestures are less frequent than “iconic” gestures, in which the form of motion is the principle communicative feature (McNeill, 1992).

**Non-textual features for topic segmentation** Research on non-textual features for topic segmentation has primarily focused on prosody, under the assumption that a key prosodic function is to mark structure at the discourse level (Steedman, 1990; Grosz and Hirshberg, 1992; Swerts, 1997). The ultimate goal of this research is to find correlates of hierarchical discourse structure in phonetic features.

Today, research on prosody has converged on prosodic cues which correlate with discourse structure. Such markers include pause duration, fundamental frequency, and pitch range manipulations (Grosz and Hirshberg, 1992; Hirschberg and Nakatani, 1998). These studies informed the development of applications such as segmentation tools for meeting analysis, e.g. (Tur et al., 2001; Galley et al., 2003).

In comparison, the connection between gesture and discourse structure is a relatively unexplored area, at least with respect to computational approaches. One conclusion that emerges from our analysis is that gesture may signal discourse structure in a different way than prosody does: while specific prosodic markers characterize segment boundaries, gesture predicts segmentation through intra-segmental cohesion. The combination of these two

modalities is an exciting direction for future research.

### 3 Visual Features for Discourse Analysis

This section describes the process of building a representation that permits the assessment of gestural cohesion. The core signal-level features are based on *spatiotemporal interest points*, which provide a sparse representation of the motion in the video. At each interest point, visual, spatial, and kinematic characteristics are extracted and then concatenated into vectors. Principal component analysis (PCA) reduces the dimensionality to a feature vector of manageable size (Bishop, 2006). These feature vectors are then clustered, yielding a codebook of visual forms. This video processing pipeline is shown in Figure 1; the remainder of the section describes the individual steps in greater detail.

#### 3.1 Spatiotemporal Interest Points

Spatiotemporal interest points (Laptev, 2005) provide a sparse representation of motion in video. The idea is to select a few local regions that contain high information content in both the spatial and temporal dimensions. The image features at these regions should be relatively robust to lighting and perspective changes, and they should capture the relevant movement in the video. The set of spatiotemporal interest points thereby provides a highly compressed representation of the key visual features. Purely spatial interest points have been successful in a variety of image processing tasks (Lowe, 1999), and spatiotemporal interest points are beginning to show similar advantages for video processing (Laptev, 2005).

The use of spatiotemporal interest points is specifically motivated by techniques from the computer vision domain of *activity recognition* (Efros et al., 2003; Niebles et al., 2006). The goal of activity recognition is to classify video sequences into semantic categories: e.g., walking, running, jumping. As a simple example, consider the task of distinguishing videos of walking from videos of jumping. In the walking videos, the motion at most of the interest points will be horizontal, while in the jumping videos it will be vertical. Spurious vertical motion in a walking video is unlikely to confuse the

classifier, as long as the majority of interest points move horizontally. The hypothesis of this paper is that just as such low-level movement features can be applied in a supervised fashion to distinguish activities, they can be applied in an unsupervised fashion to group co-speech gestures into perceptually meaningful clusters.

The Activity Recognition Toolbox (Dollár et al., 2005)<sup>1</sup> is used to detect spatiotemporal interest points for our dataset. This toolbox ranks interest points using a difference-of-Gaussians filter in the spatial dimension, and a set of Gabor filters in the temporal dimension. The total number of interest points extracted per video is set to equal the number of frames in the video. This bounds the complexity of the representation to be linear in the length of the video; however, the system may extract many interest points in some frames and none in other frames.

Figure 2 shows the interest points extracted from a representative video frame from our corpus. Note that the system has identified high contrast regions of the gesturing hand. From manual inspection, the large majority of interest points extracted in our dataset capture motion created by hand gestures. Thus, for this dataset it is reasonable to assume that an interest point-based representation expresses the visual properties of the speakers' hand gestures. In videos containing other sources of motion, preprocessing may be required to filter out interest points that are extraneous to gestural communication.

#### 3.2 Visual Descriptors

At each interest point, the temporal and spatial brightness gradients are constructed across a small space-time volume of nearby pixels. Brightness gradients have been used for a variety of problems in computer vision (Forsyth and Ponce, 2003), and provide a fairly general way to describe the visual appearance of small image patches. However, even for a small space-time volume, the resulting dimensionality is still quite large: a 10-by-10 pixel box across 5 video frames yields a 500-dimensional feature vector for each of the three gradients. For this reason, principal component analysis (Bishop, 2006) is used to reduce the dimensionality. The spatial location of the interest point is added to the final feature vector.

<sup>1</sup>[http://vision.ucsd.edu/~pdollar/research/cuboids\\_doc/index.html](http://vision.ucsd.edu/~pdollar/research/cuboids_doc/index.html)

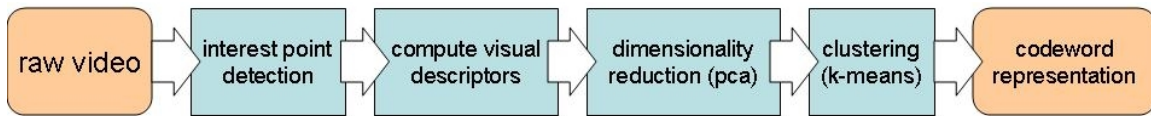


Figure 1: The visual processing pipeline for the extraction of gestural codewords from video.



Figure 2: Circles indicate the interest points extracted from this frame of the corpus.

This visual feature representation is substantially lower-level than the descriptions of gesture form found in both the psychology and computer science literatures. For example, when manually annotating gesture, it is common to employ a taxonomy of hand shapes and trajectories, and to describe the location with respect to the body and head (McNeill, 1992; Martell, 2005). Working with automatic hand tracking, Quek (2003) automatically computes perceptually-salient gesture features, such as symmetric motion and oscillatory repetitions.

In contrast, our feature representation takes the form of a vector of continuous values and is not easily interpretable in terms of how the gesture actually appears. However, this low-level approach offers several important advantages. Most critically, it requires no initialization and comparatively little tuning: it can be applied directly to any video with a fixed camera position and static background. Second, it is robust: while image noise may cause a few spurious interest points, the majority of interest points should still guide the system to an appropriate characterization of the gesture. In contrast, hand tracking can become irrevocably lost, requiring

manual resets (Gavrila, 1999). Finally, the success of similar low-level interest point representations at the activity-recognition task provides reason for optimism that they may also be applicable to unsupervised gesture analysis.

### 3.3 A Lexicon of Visual Forms

After extracting a set of low-dimensional feature vectors to characterize the visual appearance at each spatiotemporal interest point, it remains only to convert this into a representation amenable to a cohesion-based analysis. Using k-means clustering (Bishop, 2006), the feature vectors are grouped into *codewords*: a compact, lexicon-like representation of salient visual features in video. The number of clusters is a tunable parameter, though a systematic investigation of the role of this parameter is left for future work.

Codewords capture frequently-occurring patterns of motion and appearance at a local scale – interest points that are clustered together have a similar visual appearance. Because most of the motion in our videos is gestural, the codewords that appear during a given sentence provide a succinct representation of the ongoing gestural activity. Distributions of codewords over time can be analyzed in similar terms to the distribution of lexical features. A change in the distribution of codewords indicates new visual kinematic elements entering the discourse. Thus, the codeword representation allows gestural cohesion to be assessed in much the same way as lexical cohesion.

## 4 Bayesian Topic Segmentation

Topic segmentation is performed in a Bayesian framework, with each sentence’s segment index encoded in a hidden variable, written  $z_t$ . The hidden variables are assumed to be generated by a linear segmentation, such that  $z_t \in \{z_{t-1}, z_{t-1} + 1\}$ . Observations – the words and gesture codewords – are

generated by multinomial language models that are indexed according to the segment. In this framework, a high-likelihood segmentation will include language models that are tightly focused on a compact vocabulary. Such a segmentation maximizes the lexical cohesion of each segment. This model thus provides a principled, probabilistic framework for cohesion-based segmentation, and we will see that the Bayesian approach is particularly well-suited to the combination of multiple modalities.

Formally, our goal is to identify the best possible segmentation  $S$ , where  $S$  is a tuple:  $S = \langle \mathbf{z}, \theta, \phi \rangle$ . The segment indices for each sentence are written  $z_t$ ; for segment  $i$ ,  $\theta_i$  and  $\phi_i$  are multinomial language models over words and gesture codewords respectively. For each sentence,  $\mathbf{x}_t$  and  $\mathbf{y}_t$  indicate the words and gestures that appear. We will seek to identify the segmentation  $\hat{S} = \operatorname{argmax}_S p(S, \mathbf{x}, \mathbf{y})$ , conditioned on priors that will be defined below.

$$p(S, \mathbf{x}, \mathbf{y}) = p(\mathbf{x}, \mathbf{y} | S) p(S)$$

$$p(\mathbf{x}, \mathbf{y} | S) = \prod_i p(\{x_t : z_t = i\} | \theta_i) p(\{y_t : z_t = i\} | \phi_i)$$
(1)

$$p(S) = p(\mathbf{z}) \prod_i p(\theta_i) p(\phi_i)$$
(2)

The language models  $\theta_i$  and  $\phi_i$  are multinomial distributions, so the log-likelihood of the observations  $\mathbf{x}_t$  is  $\log p(\mathbf{x}_t | \theta_i) = \sum_j^W n(t, j) \log \theta_{i,j}$ , where  $n(t, j)$  is the count of word  $j$  in sentence  $t$ , and  $W$  is the size of the vocabulary. An analogous equation is used for the gesture codewords. Each language model is given a symmetric Dirichlet prior  $\alpha$ . As we will see shortly, the use of different priors for the verbal and gestural language models allows us to weight these modalities in a Bayesian framework. Finally, we model the probability of the segmentation  $\mathbf{z}$  by considering the durations of each segment:  $p(\mathbf{z}) = \prod_i p(\operatorname{dur}(i) | \psi)$ . A negative-binomial distribution with parameter  $\psi$  is applied to discourage extremely short or long segments.

**Inference** Crucially, both the likelihood (equation 1) and the prior (equation 2) factor into a product across the segments. This factorization enables the optimal segmentation to be found using a dynamic program, similar to those demonstrated by Utiyama and Isahara (2001) and Malioutov and

Barzilay (2006). For each set of segmentation points  $\mathbf{z}$ , the associated language models are set to their posterior expectations, e.g.,  $\theta_i = E[\theta | \{x_t : z_t = i\}, \alpha]$ .

The Dirichlet prior is conjugate to the multinomial, so this expectation can be computed in closed form:

$$\theta_{i,j} = \frac{n(i, j) + \alpha}{N(i) + W\alpha},$$
(3)

where  $n(i, j)$  is the count of word  $j$  in segment  $i$  and  $N(i)$  is the total number of words in segment  $i$  (Bernardo and Smith, 2000). The symmetric Dirichlet prior  $\alpha$  acts as a smoothing pseudo-count. In the multimodal context, the priors act to control the weight of each modality. If the prior for the verbal language model  $\theta$  is high relative to the prior for the gestural language model  $\phi$  then the verbal multinomial will be smoother, and will have a weaker impact on the final segmentation. The impact of the priors on the weights of each modality is explored in Section 6.

**Estimation of priors** The distribution over segment durations is negative-binomial, with parameters  $\psi$ . In general, the maximum likelihood estimate of the parameters of a negative-binomial distribution cannot be found in closed form (Balakrishnan and Nevzorov, 2003). For any given segmentation, the maximum-likelihood setting for  $\psi$  is found via a gradient-based search. This setting is then used to generate another segmentation, and the process is iterated until convergence, as in hard expectation-maximization. The Dirichlet priors on the language models are symmetric, and are chosen via cross-validation. Sampling or gradient-based techniques may be used to estimate these parameters, but this is left for future work.

**Relation to other segmentation models** Other cohesion-based techniques have typically focused on hand-crafted similarity metrics between sentences, such as cosine similarity (Galley et al., 2003; Malioutov and Barzilay, 2006). In contrast, the model described here is probabilistically motivated, maximizing the joint probability of the segmentation with the observed words and gestures. Our objective criterion is similar in form to that of Utiyama and Isahara (2001); however, in contrast to this prior

work, our criterion is justified by a Bayesian approach. Also, while the smoothing in our approach arises naturally from the symmetric Dirichlet prior, Utiyama and Isahara apply Laplace’s rule and add pseudo-counts of one in all cases. Such an approach would be incapable of flexibly balancing the contributions of each modality.

## 5 Evaluation Setup

**Dataset** Our dataset is composed of fifteen audio-video recordings of dialogues limited to three minutes in duration. The dataset includes nine different pairs of participants. In each video one of five subjects is discussed. The potential subjects include a “Tom and Jerry” cartoon, a “Star Wars” toy, and three mechanical devices: a latchbox, a piston, and a candy dispenser. One participant – “participant A” – was familiarized with the topic, and is tasked with explaining it to participant B, who is permitted to ask questions. Audio from both participants is used, but only video of participant A is used; we do not examine whether B’s gestures are relevant to discourse segmentation.

Video was recorded using standard camcorders, with a resolution of 720 by 480 at 30 frames per second. The video was reduced to 360 by 240 gray-scale images before visual analysis is applied. Audio was recorded using headset microphones. No manual postprocessing is applied to the video.

**Annotations and data processing** All speech was transcribed by hand, and time stamps were obtained using the SPHINX-II speech recognition system for forced alignment (Huang et al., 1993). Sentence boundaries are annotated according to (NIST, 2003), and additional sentence boundaries are automatically inserted at all turn boundaries. Commonly-occurring terms unlikely to impact segmentation are automatically removed by using a stoplist.

For automatic speech recognition, the default Microsoft speech recognizer was applied to each sentence, and the top-ranked recognition result was reported. As is sometimes the case in real-world applications, no speaker-specific training data is available. The resulting recognition quality is very poor, yielding a word error rate of 77%.

Annotators were instructed to select segment boundaries that divide the dialogue into coherent

topics. Segmentation points are required to coincide with sentence or turn boundaries. A second annotator – who is not an author on any paper connected with this research – provided an additional set of segment annotations on six documents. On this subset of documents, the  $P_k$  between annotators was .306, and the WindowDiff was .325 (these metrics are explained in the next subsection). This is similar to the interrater agreement reported by Malioutov and Barzilay (2006).

Over the fifteen dialogues, a total of 7458 words were transcribed (497 per dialogue), spread over 1440 sentences or interrupted turns (96 per dialogue). There were a total of 102 segments (6.8 per dialogue), from a minimum of four to a maximum of ten. This rate of fourteen sentences or interrupted turns per segment indicates relatively fine-grained segmentation. In the physics lecture corpus used by Malioutov and Barzilay (2006), there are roughly 100 sentences per segment. On the ICSI corpus of meeting transcripts, Galley *et al.* (2003) report 7.5 segments per meeting, with 770 “potential boundaries,” suggesting a similar rate of roughly 100 sentences or interrupted turns per segment.

The size of this multimodal dataset is orders of magnitude smaller than many other segmentation corpora. For example, the Broadcast News corpus used by Beeferman *et al.* (1999) and others contains two million words. The entire ICSI meeting corpus contains roughly 600,000 words, although only one third of this dataset was annotated for segmentation (Galley et al., 2003). The physics lecture corpus that was mentioned above contains 232,000 words (Malioutov and Barzilay, 2006). The task considered in this section is thus more difficult than much of the previous discourse segmentation work on two dimensions: there is less training data, and a finer-grained segmentation is required.

**Metrics** All experiments are evaluated in terms of the commonly-used  $P_k$  (Beeferman et al., 1999) and WindowDiff (WD) (Pevzner and Hearst, 2002) scores. These metrics are penalties, so lower values indicate better segmentations. The  $P_k$  metric expresses the probability that any randomly chosen pair of sentences is incorrectly segmented, if they are  $k$  sentences apart (Beeferman et al., 1999). Following tradition,  $k$  is set to half of the mean seg-

Method	$P_k$	WD
1. gesture only	.486	.502
2. ASR only	.462	.476
3. ASR + gesture	<b>.388</b>	<b>.401</b>
4. transcript only	.382	.397
5. transcript + gesture	<b>.332</b>	<b>.349</b>
6. random	.473	.526
7. equal-width	.508	.515

Table 1: For each method, the score of the best performing configuration is shown.  $P_k$  and WD are penalties, so lower values indicate better performance.

ment length. The WindowDiff metric is a variation of  $P_k$  (Pevzner and Hearst, 2002), applying a penalty whenever the number of segments within the  $k$ -sentence window differs for the reference and hypothesized segmentations.

**Baselines** Two naïve baselines are evaluated. Given that the annotator has divided the dialogue into  $K$  segments, the random baseline arbitrary chooses  $K$  random segmentation points. The results of this baseline are averaged over 1000 iterations. The equal-width baseline places boundaries such that all segments contain an equal number of sentences. Both the experimental systems and these naïve baselines were given the correct number of segments, and also were provided with manually annotated sentence boundaries – their task is to select the  $k$  sentence boundaries that most accurately segment the text.

## 6 Results

Table 1 shows the segmentation performance for a range of feature sets, as well as the two baselines. Given only gesture features the segmentation results are poor (line 1), barely outperforming the baselines (lines 6 and 7). However, gesture proves highly effective as a supplementary modality. The combination of gesture with ASR transcripts (line 3) yields an absolute 7.4% improvement over ASR transcripts alone (line 4). Paired t-tests show that this result is statistically significant ( $t(14) = 2.71, p < .01$  for both  $P_k$  and WindowDiff). Even when manual speech transcripts are available, gesture features yield a substantial improvement, reducing  $P_k$  and WD by roughly 5%. This result is statistically sig-

nificant for both  $P_k$  ( $t(14) = 2.00, p < .05$ ) and WD ( $t(14) = 1.94, p < .05$ ).

**Interactions of verbal and gesture features** We now consider the relative contribution of the verbal and gesture features. In a discriminative setting, the contribution of each modality would be explicitly weighted. In a Bayesian generative model, the same effect is achieved through the Dirichlet priors, which act to smooth the verbal and gestural multinomials – see equation 3. For example, when the gesture prior is high and verbal prior is low, the gesture counts are smoothed, and the verbal counts play a greater role in segmentation. When both priors are very high, the model will simply try to find equally-sized segments, satisfying the distribution over durations.

The effects of these parameters can be seen in Figure 3. The gesture model prior is held constant at its ideal value, and the segmentation performance is plotted against the logarithm of the verbal prior. Low values of the verbal prior cause it to dominate the segmentation; this can be seen at the left of both graphs, where the performance of the multimodal and verbal-only systems are nearly identical. High values of the verbal prior cause it to be over-smoothed, and performance thus approaches that of the gesture-only segmenter.

**Comparison to other models** While much of the research on topic segmentation focuses on written text, there are some comparable systems that also aim at unsupervised segmentation of spontaneous spoken language. For example, Malioutov and Barzilay (2006) segment a corpus of classroom lectures, using similar lexical cohesion-based features. With manual transcriptions, they report a .383  $P_k$  and .417 WD on artificial intelligence (AI) lectures, and .298  $P_k$  and .311 WD on physics lectures. Our results are in the range bracketed by these two extremes; the wide range of results suggests that segmentation scores are difficult to compare across domains. The segmentation of physics lectures was at a very coarse level of granularity, while the segmentation of AI lectures was more similar to our annotations.

We applied the publicly-available executable for this algorithm to our data, but performance was poor, yielding a .417  $P_k$  and .465 WD even when both verbal and gestural features were available.

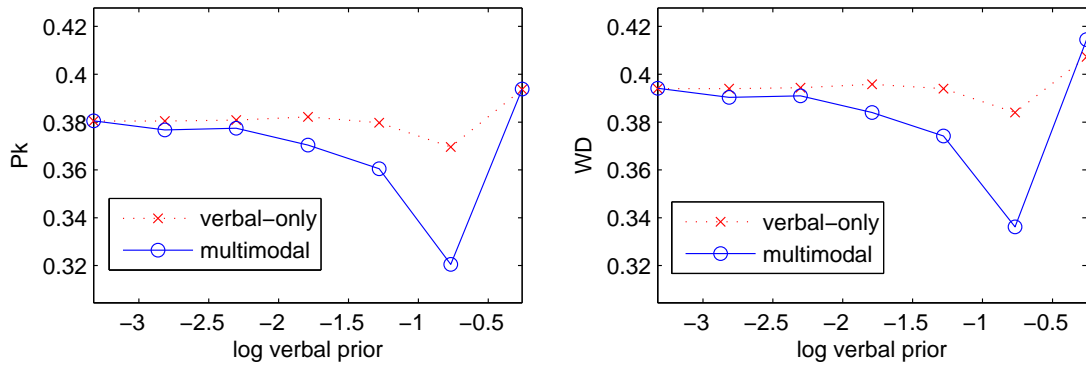


Figure 3: The multimodal and verbal-only performance using the reference transcript. The x-axis shows the logarithm of the verbal prior; the gestural prior is held fixed at the optimal value.

This may be because the technique is not designed for the relatively fine-grained segmentation demanded by our dataset (Malioutov, 2006).

## 7 Conclusions

This research shows a novel relationship between gestural cohesion and discourse structure. Automatically extracted gesture features are predictive of discourse segmentation when used in isolation; when lexical information is present, segmentation performance is further improved. This suggests that gestures provide unique information not present in the lexical features alone, even when perfect transcripts are available.

There are at least two possibilities for how gesture might impact topic segmentation: “visual punctuation,” and cohesion. The visual punctuation view would attempt to identify specific gestural patterns that are characteristic of segment boundaries. This is analogous to research that identifies prosodic signatures of topic boundaries, such as (Hirschberg and Nakatani, 1998). By design, our model is incapable of exploiting such phenomena, as our goal is to investigate the notion of gestural cohesion. Thus, the performance gains demonstrated in this paper cannot be explained by such punctuation-like phenomena; we believe that they are due to the consistent gestural themes that characterize coherent topics. However, we are interested in pursuing the idea of visual punctuation in the future, so as to compare the power of visual punctuation and gestural cohesion to predict segment boundaries. In addition, the in-

teraction of gesture and prosody suggests additional possibilities for future research.

The videos in the dataset for this paper are focused on the description of physical devices and events, leading to a fairly concrete set of gestures. In other registers of conversation, gestural form may be driven more by spatial metaphors, or may consist mainly of temporal “beats.” In such cases, the importance of gestural cohesion for discourse segmentation may depend on the visual expressivity of the speaker. We plan to examine the extensibility of gesture cohesion to more naturalistic settings, such as classroom lectures.

Finally, topic segmentation provides only an outline of the discourse structure. Richer models of discourse include hierarchical structure (Grosz and Sidner, 1986) and Rhetorical Structure Theory (Mann and Thompson, 1988). The application of gestural analysis to such models may lead to fruitful areas of future research.

## Acknowledgments

We thank Aaron Adler, C. Mario Christoudias, Michael Collins, Lisa Guttentag, Igor Malioutov, Brian Milch, Matthew Rasmussen, Candace Sidner, Luke Zettlemoyer, and the anonymous reviewers. This research was supported by Quanta Computer, the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0415865) and the Microsoft Research Faculty Fellowship.

## References

- Narayanaswamy Balakrishnan and Valery B. Nevzorov. 2003. *A primer on statistical distributions*. John Wiley & Sons.
- Doug Beeferman, Adam Berger, and John D. Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210.
- José M. Bernardo and Adrian F. M. Smith. 2000. *Bayesian Theory*. Wiley.
- Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.
- Justine Cassell, Yukiko I. Nakano, Timothy W. Bickmore, Candace L. Sidner, and Charles Rich. 2001. Non-verbal cues for discourse structure. In *Proceedings of ACL*, pages 106–115.
- Lei Chen, Mary Harper, and Zhongqiang Huang. 2006. Using maximum entropy (ME) model to incorporate gesture cues for sentence segmentation. In *Proceedings of ICMI*, pages 185–192.
- Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. 2005. Behavior recognition via sparse spatio-temporal features. In *ICCV VS-PETS*.
- Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. 2003. Recognizing action at a distance. In *Proceedings of ICCV*, pages 726–733.
- Jacob Eisenstein and Randall Davis. 2007. Conditional modality fusion for coreference resolution. In *Proceedings of ACL*, pages 352–359.
- David A. Forsyth and Jean Ponce. 2003. *Computer Vision: A Modern Approach*. Prentice Hall.
- Michel Galley, Kathleen R. McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. *Proceedings of ACL*, pages 562–569.
- Dariu M. Gavrilă. 1999. Visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98.
- Barbara Grosz and Julia Hirshberg. 1992. Some international characteristics of discourse structure. In *Proceedings of ICSLP*, pages 429–432.
- Barbara Grosz and Candace Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.
- M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.
- Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of ACL*.
- Julia Hirschberg and Christine Nakatani. 1998. Acoustic indicators of topic segmentation. In *Proceedings of ICSLP*.
- Xuedong Huang, Fileno Alleva, Mei-Yuh Hwang, and Ronald Rosenfeld. 1993. An overview of the Sphinx-II speech recognition system. In *Proceedings of ARPA Human Language Technology Workshop*, pages 81–86.
- Michael Johnston. 1998. Unification-based multimodal parsing. In *Proceedings of COLING*, pages 624–630.
- Adam Kendon. 2004. *Gesture: Visible Action as Utterance*. Cambridge University Press.
- Stefan Kopp, Paul Tepper, Kim Ferriman, and Justine Cassell. 2007. Trading spaces: How humans and humanoids use speech and gesture to give directions. In Toyoaki Nishida, editor, *Conversational Informatics: An Engineering Approach*. Wiley.
- Ivan Laptev. 2005. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123.
- David G. Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of ICCV*, volume 2, pages 1150–1157.
- Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of ACL*, pages 25–32.
- Igor Malioutov. 2006. Minimum cut model for spoken lecture segmentation. Master’s thesis, Massachusetts Institute of Technology.
- William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8:243–281.
- Craig Martell. 2005. *FORM: An experiment in the annotation of the kinematics of gesture*. Ph.D. thesis, University of Pennsylvania.
- David McNeill. 1992. *Hand and Mind*. The University of Chicago Press.
- Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. 2006. Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words. In *Proceedings of the British Machine Vision Conference*.
- NIST. 2003. The Rich Transcription Fall 2003 (RT-03F) Evaluation plan.
- Rebecca J. Passonneau and Diane J. Litman. 1997. Discourse segmentation by human and automated means. *Computational Linguistics*, 23(1):103–139.
- Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.
- Francis Quek. 2003. The catchment feature model for multimodal language analysis. In *Proceedings of ICCV*.
- Mark Steedman. 1990. Structure and intonation in spoken language understanding. In *Proceedings of ACL*, pages 9–16.
- Marc Swerts. 1997. Prosodic features at discourse boundaries of different strength. *The Journal of the Acoustical Society of America*, 101:514.
- Gokhan Tur, Dilek Hakkani-Tur, Andreas Stolcke, and Elizabeth Shriberg. 2001. Integrating prosodic and lexical cues for automatic topic segmentation. *Computational Linguistics*, 27(1):31–57.
- Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of ACL*, pages 491–498.



# Multi-Task Active Learning for Linguistic Annotations

Roi Reichart<sup>1\*</sup> Katrin Tomanek<sup>2\*</sup> Udo Hahn<sup>2</sup> Ari Rappoport<sup>1</sup>

<sup>1</sup>Institute of Computer Science  
Hebrew University of Jerusalem, Israel  
{roiri|arir}@cs.huji.ac.il

<sup>2</sup>Jena University Language & Information Engineering (JULIE) Lab  
Friedrich-Schiller-Universität Jena, Germany  
{katrin.tomanek|udo.hahn}@uni-jena.de

## Abstract

We extend the classical single-task active learning (AL) approach. In the multi-task active learning (MTAL) paradigm, we select examples for several annotation tasks rather than for a single one as usually done in the context of AL. We introduce two MTAL meta-protocols, alternating selection and rank combination, and propose a method to implement them in practice. We experiment with a two-task annotation scenario that includes named entity and syntactic parse tree annotations on three different corpora. MTAL outperforms random selection and a stronger baseline, one-sided example selection, in which one task is pursued using AL and the selected examples are provided also to the other task.

## 1 Introduction

Supervised machine learning methods have successfully been applied to many NLP tasks in the last few decades. These techniques have demonstrated their superiority over both hand-crafted rules and unsupervised learning approaches. However, they require large amounts of labeled training data for every level of linguistic processing (e.g., POS tags, parse trees, or named entities). When, when domains and text genres change (e.g., moving from common-sense newspapers to scientific biology journal articles), extensive retraining on newly supplied training material is often required, since different domains may use different syntactic structures as well as different semantic classes (entities and relations).

Consequently, with an increasing coverage of a wide variety of domains in human language technology (HLT) systems, we can expect a growing need for manual annotations to support many kinds of application-specific training data.

Creating annotated data is extremely labor-intensive. The Active Learning (AL) paradigm (Cohn et al., 1996) offers a promising solution to deal with this bottleneck, by allowing the learning algorithm to control the selection of examples to be manually annotated such that the human labeling effort be minimized. AL has been successfully applied already for a wide range of NLP tasks, including POS tagging (Engelson and Dagan, 1996), chunking (Ngai and Yarowsky, 2000), statistical parsing (Hwa, 2004), and named entity recognition (Tomanek et al., 2007).

However, AL is designed in such a way that it selects examples for manual annotation with respect to a *single* learning algorithm or classifier. Under this AL annotation policy, one has to perform a separate annotation cycle for each classifier to be trained. In the following, we will refer to the annotations supplied for a classifier as the annotations for a *single annotation task*.

Modern HLT systems often utilize annotations resulting from different tasks. For example, a machine translation system might use features extracted from parse trees and named entity annotations. For such an application, we obviously need the different annotations to reside in the same text corpus. It is not clear how to apply the single-task AL approach here, since a training example that is beneficial for one task might not be so for others. We could annotate

\* Both authors contributed equally to this work.

the same corpus independently by the two tasks and merge the resulting annotations, but that (as we show in this paper) would possibly yield sub-optimal usage of human annotation efforts.

There are two reasons why multi-task AL, and by this, a combined corpus annotated for various tasks, could be of immediate benefit. First, annotators working on *similar* annotation tasks (e.g., considering named entities and relations between them), might exploit annotation data from one subtask for the benefit of the other. If for each subtask a separate corpus is sampled by means of AL, annotators will definitely lack synergy effects and, therefore, annotation will be more laborious and is likely to suffer in terms of quality and accuracy. Second, for *dissimilar* annotation tasks – take, e.g., a comprehensive HLT pipeline incorporating morphological, syntactic and semantic data – a classifier might require features as input which constitute the output of another preceding classifier. As a consequence, training such a classifier which takes into account several annotation tasks will best be performed on a rich corpus annotated with respect to all input-relevant tasks. Both kinds of annotation tasks, similar and dissimilar ones, constitute examples of what we refer to as *multi-task* annotation problems.

Indeed, there have been efforts in creating resources annotated with respect to various annotation tasks though each of them was carried out independently of the other. In the general language UPenn annotation efforts for the WSJ sections of the Penn Treebank (Marcus et al., 1993), sentences are annotated with POS tags, parse trees, as well as discourse annotation from the Penn Discourse Treebank (Mitsakaki et al., 2008), while verbs and verb arguments are annotated with Propbank rolesets (Palmer et al., 2005). In the biomedical GENIA corpus (Ohta et al., 2002), scientific text is annotated with POS tags, parse trees, and named entities.

In this paper, we introduce *multi-task active learning* (MTAL), an active learning paradigm for multiple annotation tasks. We propose a new AL framework where the examples to be annotated are selected so that they are as informative as possible for a *set* of classifiers instead of a single classifier only. This enables the creation of a single combined corpus annotated with respect to various annotation tasks, while preserving the advantages of AL with

respect to the minimization of annotation efforts.

In a proof-of-concept scenario, we focus on two highly dissimilar tasks, syntactic parsing and named entity recognition, study the effects of multi-task AL under rather extreme conditions. We propose two MTAL meta-protocols and a method to implement them for these tasks. We run experiments on three corpora for domains and genres that are very different (WSJ: newspapers, Brown: mixed genres, and GENIA: biomedical abstracts). Our protocols outperform two baselines (random and a stronger one-sided selection baseline).

In Section 2 we introduce our MTAL framework and present two MTAL protocols. In Section 3 we discuss the evaluation of these protocols. Section 4 describes the experimental setup, and results are presented in Section 5. We discuss related work in Section 6. Finally, we point to open research issues for this new approach in Section 7.

## 2 A Framework for Multi-Task AL

In this section we introduce a sample selection framework that aims at reducing the human annotation effort in a multiple annotation scenario.

### 2.1 Task Definition

To measure the efficiency of selection methods, we define the *training quality*  $TQ$  of annotated material  $S$  as the performance  $p$  yielded with a reference learner  $X$  trained on that material:  $TQ(X, S) = p$ . A selection method can be considered better than another one if a higher TQ is yielded with the same amount of examples being annotated.

Our framework is an extension of the Active Learning (AL) framework (Cohn et al., 1996). The original AL framework is based on querying in an iterative manner those examples to be manually annotated that are most useful for the learner at hand. The TQ of an annotated corpus selected by means of AL is much higher than random selection. This AL approach can be considered as *single-task AL* because it focuses on a single learner for which the examples are to be selected. In a multiple annotation scenario, however, there are several annotation tasks to be accomplished at once and for each task typically a separate statistical model will then be trained. Thus, the goal of *multi-task AL* is to query those examples for

human annotation that are most informative for *all* learners involved.

## 2.2 One-Sided Selection vs. Multi-Task AL

The naive approach to select examples in a multiple annotation scenario would be to perform a single-task AL selection, i.e., the examples to be annotated are selected with respect to one of the learners only.<sup>1</sup> In a multiple annotation scenario we call such an approach *one-sided* selection. It is an *intrinsic* selection for the reference learner, and an *extrinsic* selection for all the other learners also trained on the annotated material. Obviously, a corpus compiled with the help of one-sided selection will have a good TQ for that learner for which the intrinsic selection has taken place. For all the other learners, however, we have no guarantee that their TQ will not be inferior than the TQ of a random selection process.

In scenarios where the different annotation tasks are highly dissimilar we can expect extrinsic selection to be rather poor. This intuition is demonstrated by experiments we conducted for named entity (NE) and parse annotation tasks<sup>2</sup> (Figure 1). In this scenario, extrinsic selection for the NE annotation task means that examples were selected with respect to the parsing task. Extrinsic selection performed about the same as random selection for the NE task, while for the parsing task extrinsic selection performed markedly worse. This shows that examples that were very informative for the NE learner were not that informative for the parse learner.

## 2.3 Protocols for Multi-Task AL

Obviously, we can expect one-sided selection to perform better for the reference learner (the one for which an intrinsic selection took place) than multi-task AL selection, because the latter would be a compromise for all learners involved in the multiple annotation scenario. However, the goal of multi-task AL is to minimize the annotation effort over all annotation tasks and not just the effort for a single annotation task.

For a multi-task AL protocol to be valuable in a specific multiple annotation scenario, the TQ for all considered learners should be

<sup>1</sup>Of course, all selected examples would be annotated w.r.t. all annotation tasks.

<sup>2</sup>See Section 4 for our experimental setup.

1. better than the TQ of random selection,
2. and better than the TQ of any extrinsic selection.

In the following, we introduce two protocols for multi-task AL. Multi-task AL protocols can be considered *meta-protocols* because they basically specify how task-specific, single-task AL approaches can be combined into one selection decision. By this, the protocols are independent of the underlying task-specific AL approaches.

### 2.3.1 Alternating Selection

The *alternating selection* protocol alternates one-sided AL selection. In  $s_j$  consecutive AL iterations, the selection is performed as one-sided selection with respect to learning algorithm  $X_j$ . After that, another learning algorithm is considered for selection for  $s_k$  consecutive iterations and so on. Depending on the specific scenario, this enables to weight the different annotation tasks by allowing them to guide the selection in more or less AL iterations. This protocol is a straight-forward compromise between the different single-task selection approaches.

In this paper we experiment with the special case of  $s_i = 1$ , where in every AL iteration the selection leadership is changed. More sophisticated calibration of the parameters  $s_i$  is beyond the scope of this paper and will be dealt with in future work.

### 2.3.2 Rank Combination

The *rank combination* protocol is more directly based on the idea to combine single-task AL selection decisions. In each AL iteration, the usefulness score  $s_{X_j}(e)$  of each unlabeled example  $e$  from the pool of examples is calculated with respect to each learner  $X_j$  and then translated into a rank  $r_{X_j}(e)$  where higher usefulness means lower rank number (examples with identical scores get the same rank number). Then, for each example, we sum the rank numbers of each annotation task to get the overall rank  $r(e) = \sum_{j=1}^n r_{X_j}(e)$ . All examples are sorted by this combined rank and  $b$  examples with lowest rank numbers are selected for manual annotation.<sup>3</sup>

<sup>3</sup>As the number of ranks might differ between the single annotation tasks, we normalize them to the coarsest scale. Then we can sum up the ranks as explained above.

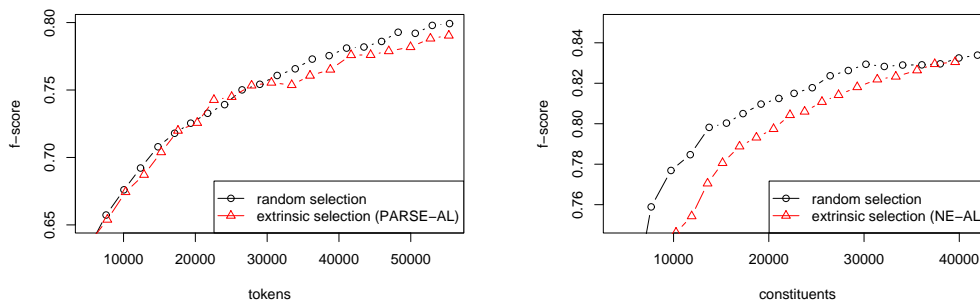


Figure 1: Learning curves for random and extrinsic selection on both tasks: named entity annotation (left) and syntactic parse annotation (right), using the WSJ corpus scenario

This protocol favors examples which are good for all learning algorithms. Examples that are highly informative for one task but rather uninformative for another task will not be selected.

### 3 Evaluation of Multi-Task AL

The notion of training quality (TQ) can be used to quantify the effectiveness of a protocol, and by this, annotation costs in a single-task AL scenario. To actually quantify the overall training quality in a multiple annotation scenario one would have to sum over all the single task’s TQs. Of course, depending on the specific annotation task, one would not want to quantify the number of examples being annotated but different task-specific units of annotation. While for entity annotations one does typically count the number of tokens being annotated, in the parsing scenario the number of constituents being annotated is a generally accepted measure. As, however, the actual time needed for the annotation of one example usually differs for different annotation tasks, normalizing exchange rates have to be specified which can then be used as weighting factors. In this paper, we do not define such weighting factors<sup>4</sup>, and leave this challenging question to be discussed in the context of psycholinguistic research.

We could quantify the overall efficiency score  $E$  of a MTAL protocol  $P$  by

$$E(P) = \sum_{j=1}^n \alpha_j \cdot TQ(X_j, u_j)$$

where  $u_j$  denotes the individual annotation task’s

<sup>4</sup>Such weighting factors not only depend on the annotation level or task but also on the domain, and especially on the cognitive load of the annotation task.

number of units being annotated (e.g., constituents for parsing) and the task-specific weights are defined by  $\alpha_j$ . Given weights are properly defined, such a score can be applied to directly compare different protocols and quantify their differences.

In practice, such task-specific weights might also be considered in the MTAL protocols. In the alternating selection protocol, the numbers of consecutive iterations  $s_i$  each single task protocol can be tuned according to the  $\alpha$  parameters. As for the rank combination protocol, the weights can be considered when calculating the overall rank:  $r(e) = \sum_{j=1}^n \beta_j \cdot r_{X_j}(e)$  where the parameters  $\beta_1 \dots \beta_n$  reflect the values of  $\alpha_1 \dots \alpha_n$  (though they need not necessarily be the same).

In our experiments, we assumed the same weight for all annotation schemata, thus simply setting  $s_i = 1, \beta_i = 1$ . This was done for the sake of a clear framework presentation. Finding proper weights for the single tasks and tuning the protocols accordingly is a subject for further research.

## 4 Experiments

### 4.1 Scenario and Task-Specific Selection Protocols

The tasks in our scenario comprise one semantic task (annotation with named entities (NE)) and one syntactic task (annotation with PCFG parse trees). The tasks are highly dissimilar, thus increasing the potential value of MTAL. Both tasks are subject to intensive research by the NLP community.

The MTAL protocols proposed are meta-protocols that combine the selection decisions of the underlying, task-specific AL protocols. In our scenario, the task-specific AL protocols are

committee-based (Freund et al., 1997) selection protocols. In committee-based AL, a committee consists of  $k$  classifiers of the same type trained on different subsets of the training data.<sup>5</sup> Each committee member then makes its predictions on the unlabeled examples, and those examples on which the committee members disagree most are considered most informative for learning and are thus selected for manual annotation. In our scenario the example grain-size is the sentence level.

For the NE task, we apply the AL approach of Tomanek et al. (2007). The committee consists of  $k_1 = 3$  classifiers and the vote entropy (VE) (Engelson and Dagan, 1996) is employed as disagreement metric. It is calculated on the token-level as

$$VE_{tok}(t) = -\frac{1}{\log k} \sum_{i=0}^c \frac{V(l_i, t)}{k} \log \frac{V(l_i, t)}{k} \quad (1)$$

where  $\frac{V(l_i, t)}{k}$  is the ratio of  $k$  classifiers where the label  $l_i$  is assigned to a token  $t$ . The sentence level vote entropy  $VE_{sent}$  is then the average over all tokens  $t_j$  of sentence  $s$ .

For the parsing task, the disagreement score is based on a committee of  $k_2 = 10$  instances of Dan Bikel’s reimplementation of Collins’ parser (Bickel, 2005; Collins, 1999). For each sentence in the unlabeled pool, the agreement between the committee members was calculated using the function reported by Reichart and Rappoport (2007):

$$AF(s) = \frac{1}{N} \sum_{i, l \in [1 \dots N], i \neq l} f_{score}(m_i, m_l) \quad (2)$$

Where  $m_i$  and  $m_l$  are the committee members and  $N = \frac{k_2 \cdot (k_2 - 1)}{2}$  is the number of pairs of different committee members. This function calculates the agreement between the members of each pair by calculating their relative f-score and then averages the pairs’ scores. The disagreement of the committee on a sentence is simply  $1 - AF(s)$ .

## 4.2 Experimental settings

For the NE task we employed the classifier described by Tomanek et al. (2007): The NE tagger is based on Conditional Random Fields (Lafferty et al., 2001)

<sup>5</sup>We randomly sampled  $L = \frac{3}{4}$  of the training data to create each committee member.

and has a rich feature set including orthographical, lexical, morphological, POS, and contextual features. For parsing, Dan Bikel’s reimplementation of Collins’ parser is employed, using gold POS tags.

In each AL iteration we select 100 sentences for manual annotation.<sup>6</sup> We start with a randomly chosen seed set of 200 sentences. Within a corpus we used the same seed set in all selection scenarios. We compare the following five selection scenarios: Random selection (*RS*), which serves as our baseline; one-sided AL selection for both tasks (called *NE-AL* and *PARSE-AL*); and multi-task AL selection with the alternating selection protocol (*alter-MTAL*) and the rank combination protocol (*ranks-MTAL*).

We performed our experiments on three different corpora, namely one from the newspaper genre (*WSJ*), a mixed-genre corpus (*Brown*), and a biomedical corpus (*Bio*). Our simulation corpora contain both entity annotations and (constituent) parse annotations. For each corpus we have a pool set (from which we select the examples for annotation) and an evaluation set (used for generating the learning curves). The *WSJ* corpus is based on the *WSJ* part of the *PENN TREEBANK* (Marcus et al., 1993); we used the first 10,000 sentences of section 2-21 as the pool set, and section 00 as evaluation set (1,921 sentences). The *Brown* corpus is also based on the respective part of the *PENN TREEBANK*. We created a sample consisting of 8 of any 10 consecutive sentences in the corpus. This was done as *Brown* contains text from various English text genres, and we did that to create a representative sample of the corpus domains. We finally selected the first 10,000 sentences from this sample as pool set. Every 9th from every 10 consecutive sentences package went into the evaluation set which consists of 2,424 sentences. For both *WSJ* and *Brown* only parse annotations though no entity annotations were available. Thus, we enriched both corpora with entity annotations (three entities: person, location, and organization) by means of a tagger trained on the English data set of the *CoNLL-2003* shared task (Tjong Kim Sang and De Meulder, 2003).<sup>7</sup> The *Bio* corpus

<sup>6</sup>Manual annotation is simulated by just unveiling the annotations already contained in our corpora.

<sup>7</sup>We employed a tagger similar to the one presented by Settles (2004). Our tagger has a performance of  $\approx 84\%$  f-score on the *CoNLL-2003* data; inspection of the predicted entities on

is based on the parsed section of the GENIA corpus (Ohta et al., 2002). We performed the same divisions as for Brown, resulting in 2,213 sentences in our pool set and 276 sentences for the evaluation set. This part of the GENIA corpus comes with entity annotations. We have collapsed the entity classes annotated in GENIA (cell line, cell type, DNA, RNA, protein) into a single, biological entity class.

## 5 Results

In this section we present and discuss our results when applying the five selection strategies (RS, NE-AL, PARSE-AL, alter-MTAL, and ranks-MTAL) to our scenario on the three corpora. We refrain from calculating the overall efficiency score (Section 3) here due to the lack of generally accepted weights for the considered annotation tasks. However, we require from a good selection protocol to exceed the performance of random selection and extrinsic selection. In addition, recall from Section 3 that we set the alternate selection and rank combination parameters to  $s_i = 1, \beta_i = 1$ , respectively to reflect a tradeoff between the annotation efforts of both tasks.

Figures 2 and 3 depict the learning curves for the NE tagger and the parser on WSJ and Brown, respectively. Each figure shows the five selection strategies. As expected, on both corpora and both tasks intrinsic selection performs best, i.e., for the NE tagger NE-AL and for the parser PARSE-AL. Further, random selection and extrinsic selection perform worst. Most importantly, both MTAL protocols clearly outperform extrinsic and random selection in all our experiments. This is in contrast to NE-AL which performs worse than random selection for all corpora when used as extrinsic selection, and for PARSE-AL that outperforms the random baseline only for Brown when used as extrinsic selection. That is, the MTAL protocols suggest a tradeoff between the annotation efforts of the different tasks, here.

On WSJ, both for the NE and the parse annotation tasks, the performance of the MTAL protocols is very similar, though ranks-MTAL performs slightly better. For the parser task, up to 30,000 constituents MTAL performs almost as good as does PARSE-AL. This is different for the NE task where NE-AL

clearly outperforms MTAL. On Brown, in general we see the same results, with some minor differences. On the NE task, extrinsic selection (PARSE-AL) performs better than random selection, but it is still much worse than intrinsic AL or MTAL. Here, ranks-MTAL significantly outperforms alter-MTAL and almost performs as good as intrinsic selection. For the parser task, we see that extrinsic and random selection are equally bad. Both MTAL protocols perform equally well, again being quite similar to the intrinsic selection. On the BIO corpus<sup>8</sup> we observed the same tendencies as in the other two corpora, i.e., MTAL clearly outperforms extrinsic and random selection and supplies a better tradeoff between annotation efforts of the task at hand than one-sided selection.

Overall, we can say that in all scenarios MTAL performs much better than random selection and extrinsic selection, and in most cases the performance of MTAL (especially but not exclusively, ranks-MTAL) is even close to intrinsic selection. This is promising evidence that MTAL selection can be a better choice than one-sided selection in multiple annotation scenarios. Thus, considering all annotation tasks in the selection process (even if the selection protocol is as simple as the alternating selection protocol) is better than selecting only with respect to one task. Further, it should be noted that overall the more sophisticated rank combination protocol does not perform much better than the simpler alternating selection protocol in all scenarios.

Finally, Figure 4 shows the disagreement curves for the two tasks on the WSJ corpus. As has already been discussed by Tomanek and Hahn (2008), disagreement curves can be used as a stopping criterion and to monitor the progress of AL-driven annotation. This is especially valuable when no annotated validation set is available (which is needed for plotting learning curves). We can see that the disagreement curves significantly flatten approximately at the same time as the learning curves do. In the context of MTAL, disagreement curves might not only be interesting as a stopping criterion but rather as a switching criterion, i.e., to identify when MTAL could be turned into one-sided selection. This would be the case if in an MTAL scenario, the disagree-

WSJ and Brown revealed a good tagging performance.

<sup>8</sup>The plots for the Bio are omitted due to space restrictions.

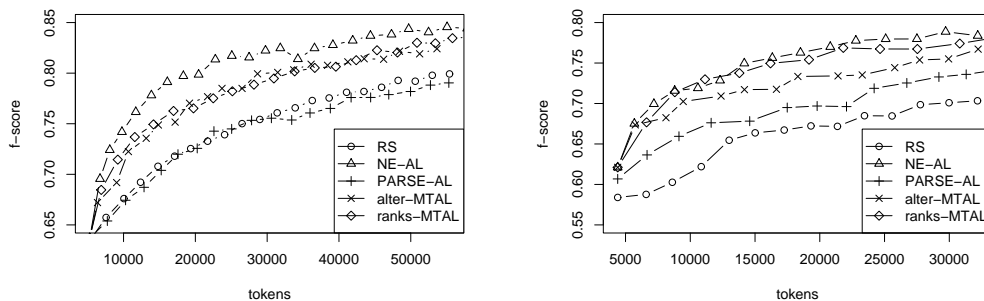


Figure 2: Learning curves for NE task on WSJ (left) and Brown (right)

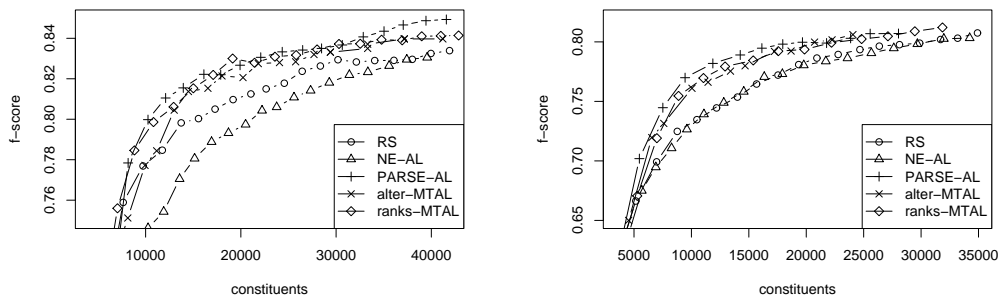


Figure 3: Learning curves for parse task on WSJ (left) and Brown (right)

ment curve of one task has a slope of (close to) zero. Future work will focus on issues related to this.

## 6 Related Work

There is a large body of work on single-task AL approaches for many NLP tasks where the focus is mainly on better, task-specific selection protocols and methods to quantify the usefulness score in different scenarios. As to the tasks involved in our scenario, several papers address AL for NER (Shen et al., 2004; Hachey et al., 2005; Tomanek et al., 2007) and syntactic parsing (Tang et al., 2001; Hwa, 2004; Baldridge and Osborne, 2004; Becker and Osborne, 2005). Further, there is some work on questions arising when AL is to be used in real-life annotation scenarios, including impaired inter-annotator agreement, stopping criteria for AL-driven annotation, and issues of reusability (Baldridge and Osborne, 2004; Hachey et al., 2005; Zhu and Hovy, 2007; Tomanek et al., 2007).

Multi-task AL is methodologically related to approaches of decision combination, especially in the context of classifier combination (Ho et al., 1994) and ensemble methods (Breiman, 1996). Those approaches focus on the combination of classifiers in

order to improve the classification error rate for one specific classification task. In contrast, the focus of multi-task AL is on strategies to select training material for multi classifier systems where all classifiers cover different classification tasks.

## 7 Discussion

Our treatment of MTAL within the context of the orthogonal two-task scenario leads to further interesting research questions. First, future investigations will have to focus on the question whether the positive results observed in our orthogonal (i.e., highly dissimilar) two-task scenario will also hold for a more realistic (and maybe more complex) multiple annotation scenario where tasks are more similar and more than two annotation tasks might be involved. Furthermore, several forms of *interdependencies* may arise between the single annotation tasks. As a first example, consider the (functional) interdependencies (i.e., task similarity) in higher-level semantic NLP tasks of relation or event recognition. In such a scenario, several tasks including entity annotations and relation/event annotations, as well as syntactic parse data, have to be incorporated at the same time. Another type of (data flow) inter-

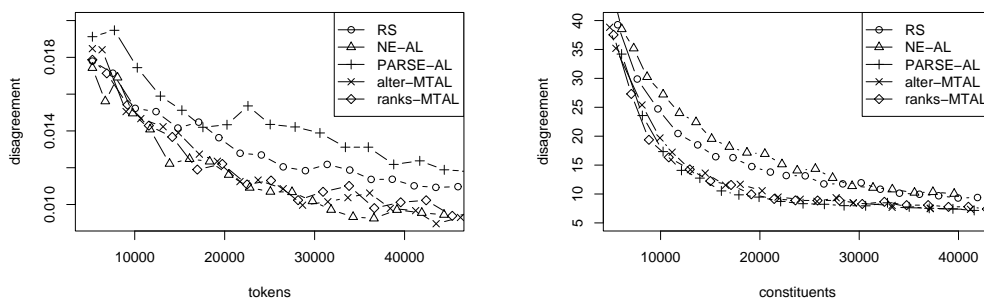


Figure 4: Disagreement curves for NE task (left) and parse task (right) on WSJ

dependency occurs in a second scenario where material for several classifiers that are data-dependent on each other – one takes the output of another classifier as input features – has to be efficiently annotated. Whether the proposed protocols are beneficial in the context of such highly interdependent tasks is an open issue. Even more challenging is the idea to provide methodologies helping to predict in an arbitrary application scenario whether the choice of MTAL is truly advantageous.

Another open question is how to measure and quantify the overall *annotation costs* in multiple annotation scenarios. Exchange rates are inherently tied to the specific task and domain. In practice, one might just want to measure the time needed for the annotations. However, in a simulation scenario, a common metric is necessary to compare the performance of different selection strategies with respect to the overall annotation costs. This requires studies on how to quantify, with a comparable cost function, the efforts needed for the annotation of a textual unit of choice (e.g., tokens, sentences) with respect to different annotation tasks.

Finally, the question of *reusability* of the annotated material is an important issue. Reusability in the context of AL means to which degree corpora assembled with the help of any AL technique can be (re)used as a general resource, i.e., whether they are well suited for the training of classifiers other than the ones used during the selection process. This is especially interesting as the details of the classifiers that should be trained in a later stage are typically not known at the resource building time. Thus, we want to select samples valuable to a *family* of classifiers using the various annotation layers. This, of course, is only possible if data annotated with the

help of AL is reusable by modified though similar classifiers (e.g., with respect to the features being used) – compared to the classifiers employed for the selection procedure.

The issue of reusability has already been raised but not yet conclusively answered in the context of single-task AL (see Section 6). Evidence was found that reusability up to a certain, though not well-specified, level is possible. Of course, reusability has to be analyzed separately in the context of various MTAL scenarios. We feel that these scenarios might both be more challenging and more relevant to the reusability issue than the single-task AL scenario, since resources annotated with multiple layers can be used to the design of a larger number of a (possibly more complex) learning algorithms.

## 8 Conclusions

We proposed an extension to the single-task AL approach such that it can be used to select examples for annotation with respect to several annotation tasks. To the best of our knowledge this is the first paper on this issue, with a focus on NLP tasks. We outlined a problem definition and described a framework for multi-task AL. We presented and tested two protocols for multi-task AL. Our results are promising as they give evidence that in a multiple annotation scenario, multi-task AL outperforms naive one-sided and random selection.

## Acknowledgments

The work of the second author was funded by the German Ministry of Education and Research within the STEMNET project (01DS001A-C), while the work of the third author was funded by the EC within the BOOTSTREP project (FP6-028099).



## References

- Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of EMNLP'04*, pages 9–16.
- Markus Becker and Miles Osborne. 2005. A two-stage method for active learning of statistical grammars. In *Proceedings of IJCAI'05*, pages 991–996.
- Daniel M. Bickel. 2005. Code developed at the University of Pennsylvania, <http://www.cis.upenn.edu/~dbikel/software.html>.
- Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.
- David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.
- Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.
- Sean Engelson and Ido Dagan. 1996. Minimizing manual annotation cost in supervised training from corpora. In *Proceedings of ACL'96*, pages 319–326.
- Yoav Freund, Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168.
- Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *Proceedings of CoNLL'05*, pages 144–151.
- Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. 1994. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75.
- Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.
- John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML'01*, pages 282–289.
- 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.
- Eleni Miltsakaki, Livio Robaldo, Alan Lee, and Aravind K. Joshi. 2008. Sense annotation in the penn discourse treebank. In *Proceedings of CICLing'08*, pages 275–286.
- Grace Ngai and David Yarowsky. 2000. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *Proceedings of ACL'00*, pages 117–125.
- Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of HLT'02*, pages 82–86.
- Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.
- Roi Reichart and Ari Rappoport. 2007. An ensemble method for selection of high quality parses. In *Proceedings of ACL'07*, pages 408–415, June.
- Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of JNLPBA'04*, pages 107–110.
- Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of ACL'04*, pages 589–596.
- Min Tang, Xiaoqiang Luo, and Salim Roukos. 2001. Active learning for statistical natural language parsing. In *Proceedings of ACL'02*, pages 120–127.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL'03*, pages 142–147.
- Katrin Tomanek and Udo Hahn. 2008. Approximating learning curves for active-learning-driven annotation. In *Proceedings of LREC'08*.
- Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains corpus reusability of annotated data. In *Proceedings of EMNLP-CoNLL'07*, pages 486–495.
- Jingbo Zhu and Eduard Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of EMNLP-CoNLL'07*, pages 783–790.

# Generalized Expectation Criteria for Semi-Supervised Learning of Conditional Random Fields

Gideon S. Mann

Google Inc.  
76 Ninth Avenue  
New York, NY 10011

Andrew McCallum

Department of Computer Science  
University of Massachusetts  
140 Governors Drive  
Amherst, MA 01003

## Abstract

This paper presents a semi-supervised training method for linear-chain conditional random fields that makes use of labeled features rather than labeled instances. This is accomplished by using *generalized expectation* criteria to express a preference for parameter settings in which the model's distribution on unlabeled data matches a target distribution. We induce target conditional probability distributions of labels given features from both annotated feature occurrences in context and ad-hoc feature majority label assignment. The use of generalized expectation criteria allows for a dramatic reduction in annotation time by shifting from traditional instance-labeling to feature-labeling, and the methods presented outperform traditional CRF training and other semi-supervised methods when limited human effort is available.

## 1 Introduction

A significant barrier to applying machine learning to new real world domains is the cost of obtaining the necessary training data. To address this problem, work over the past several years has explored semi-supervised or unsupervised approaches to the same problems, seeking to improve accuracy with the addition of lower cost unlabeled data. Traditional approaches to semi-supervised learning are applied to cases in which there is a small amount of fully labeled data and a much larger amount of unlabeled data, presumably from the same data source. For example, EM (Nigam et al., 1998), transductive SVMs (Joachims, 1999), entropy regularization (Grandvalet and Bengio, 2004), and graph-based

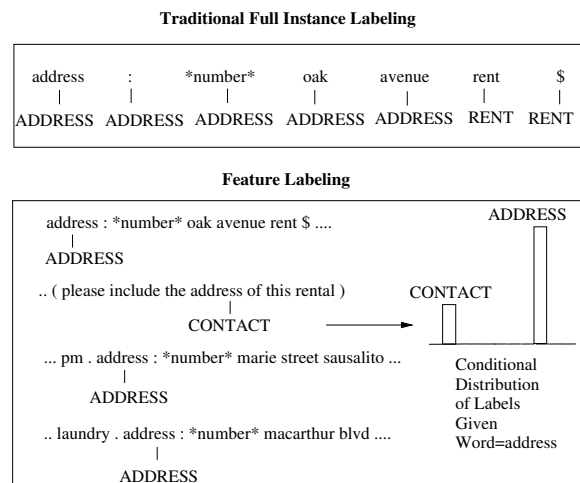


Figure 1: Top: Traditional instance-labeling in which sequences of contiguous tokens are annotated as to their correct label. Bottom: Feature-labeling in which non-contiguous feature occurrences in context are labeled for the purpose of deriving a conditional probability distribution of labels given a particular feature.

methods (Zhu and Ghahramani, 2002; Szummer and Jaakkola, 2002) have all been applied to a limited amount of fully labeled data in conjunction with unlabeled data to improve the accuracy of a classifier.

In this paper, we explore an alternative approach in which, instead of fully labeled instances, the learner has access to *labeled features*. These features can often be labeled at a lower-cost to the human annotator than labeling entire instances, which may require annotating the multiple sub-parts of a sequence structure or tree. Features can be labeled either by specifying the majority label for a particular feature or by annotating a few occurrences of a particular feature in context with the correct label (Figure 1).

To train models using this information we use

generalized expectation (GE) criteria. GE criteria are terms in a training objective function that assign scores to values of a model expectation. In particular we use a version of GE that prefers parameter settings in which certain model expectations are close to target distributions. Previous work has shown how to apply GE criteria to maximum entropy classifiers. In section 4, we extend GE criteria to semi-supervised learning of linear-chain conditional random fields, using conditional probability distributions of labels given features.

To empirically evaluate this method we compare it with several competing methods for CRF training, including entropy regularization and expected gradient, showing that GE provides significant improvements. We achieve competitive performance in comparison to alternate model families, in particular generative models such as MRFs trained with EM (Haghighi and Klein, 2006) and HMMs trained with soft constraints (Chang et al., 2007). Finally, in Section 5.3 we show that feature-labeling can lead to dramatic reductions in the annotation time that is required in order to achieve the same level of accuracy as traditional instance-labeling.

## 2 Related Work

There has been a significant amount of work on semi-supervised learning with small amounts of *fully labeled* data (see Zhu (2005)). However there has been comparatively less work on learning from alternative forms of labeled resources. One example is Schapire et al. (2002) who present a method in which features are annotated with their associated majority labels and this information is used to bootstrap a parameterized text classification model. Unlike the model presented in this paper, they require some labeled data in order to train their model.

This type of input information (features + majority label) is a powerful and flexible model for specifying alternative inputs to a classifier, and has been additionally used by Haghighi and Klein (2006). In that work, “prototype” features—words with their associated labels—are used to train a generative MRF sequence model. Their probability model can be formally described as:

$$p_{\theta}(\mathbf{x}, \mathbf{y}) = \frac{1}{Z(\theta)} \exp \left( \sum_k \theta_k F_k(\mathbf{x}, \mathbf{y}) \right).$$

Although the partition function must be computed over all  $(\mathbf{x}, \mathbf{y})$  tuples, learning via EM in this model is possible because of approximations made in computing the partition function.

Another way to gather supervision is by means of prior label distributions. Mann and McCallum (2007) introduce a special case of GE, *label regularization*, and demonstrate its effectiveness for training maximum entropy classifiers. In label regularization, the model prefers parameter settings in which the model’s predicted label distribution on the unsupervised data match a target distribution. Note that supervision here consists of the the full distribution over labels (i.e. conditioned on the maximum entropy “default feature”), instead of simply the majority label. Druck et al. (2007) also use GE with full distributions for semi-supervised learning of maximum entropy models, except here the distributions are on labels *conditioned on features*. In Section 4 we describe how GE criteria can be applied to CRFs given conditional probability distributions of labels given features.

Another recent method that has been proposed for training sequence models with constraints is Chang et al. (2007). They use constraints for approximate EM training of an HMM, incorporating the constraints by looking only at the top  $K$  most-likely sequences from a joint model of likelihood and the constraints. This model can be applied to the combination of labeled and unlabeled instances, but cannot be applied in situations where only labeled features are available. Additionally, our model can be easily combined with other semi-supervised criteria, such as entropy regularization. Finally, their model is a generative HMM which cannot handle the rich, non-independent feature sets that are available to a CRF.

There have been relatively few different approaches to CRF semi-supervised training. One approach has been that proposed in both Miller et al. (2004) and Freitag (2004), uses distributional clustering to induce features from a large corpus, and then uses these features to augment the feature space of the labeled data. Since this is an orthogonal method for improving accuracy it can be combined with many of the other methods discussed above, and indeed we have obtained positive preliminary experimental results with GE criteria (not reported on here).

Another method for semi-supervised CRF training is entropy regularization, initially proposed by Grandvalet and Bengio (2004) and extended to linear-chain CRFs by Jiao et al. (2006). In this formulation, the traditional label likelihood (on supervised data) is augmented with an additional term that encourages the model to predict low-entropy label distributions on the unlabeled data:

$$\mathcal{O}(\theta; D, U) = \sum_d \log p_\theta(\mathbf{y}^{(d)} | \mathbf{x}^{(d)}) - \lambda H(\mathbf{y} | \mathbf{x}).$$

This method can be quite brittle, since the minimal entropy solution assigns all of the tokens the same label.<sup>1</sup> In general, entropy regularization is fragile, and accuracy gains can come only with precise settings of  $\lambda$ . High values of  $\lambda$  fall into the minimal entropy trap, while low values of  $\lambda$  have no effect on the model (see (Jiao et al., 2006) for an example).

When some instances have partial labelings (i.e. labels for some of their tokens), it is possible to train CRFs via expected gradient methods (Salakhutdinov et al., 2003). Here a reformulation is presented in which the gradient is computed for a probability distribution with a marginalized hidden variable,  $z$ , and observed training labels  $y$ :

$$\begin{aligned} \nabla_L(\theta) &= \frac{\partial}{\partial \theta} \sum_z \log p(x, y, z; \theta) \\ &= \sum_z p(z|y, x) f_k(x, y, z) \\ &\quad - \sum_{z, y'} p(z, y' | x; \theta) f_k(x, y, z). \end{aligned}$$

In essence, this resembles the standard gradient for the CRF, except that there is an additional marginalization in the first term over the hidden variable  $z$ . This type of training has been applied by Quattoni et al. (2007) for hidden-state conditional random fields, and can be equally applied to semi-supervised conditional random fields. Note, however, that labeling variables of a structured instance (e.g. tokens) is different than labeling features—being both more coarse-grained and applying supervision narrowly only to the individual subpart, not to all places in the data where the feature occurs.

<sup>1</sup>In the experiments in this paper, we use  $\lambda = 0.001$ , which we tuned for best performance on the test set, giving an unfair advantage to our competitor.

Finally, there are some methods that use auxiliary tasks for training sequence models, though they do not train linear-chain CRFs per se. Ando and Zhang (2005) include a cluster discovery step into the supervised training. Smith and Eisner (2005) use neighborhoods of related instances to figure out what makes found instances “good”. Although these methods can often find good solutions, both are quite sensitive to the selection of auxiliary information, and making good selections requires significant insight.<sup>2</sup>

### 3 Conditional Random Fields

Linear-chain conditional random fields (CRFs) are a discriminative probabilistic model over sequences  $\mathbf{x}$  of feature vectors and label sequences  $\mathbf{y} = \langle y_1..y_n \rangle$ , where  $|\mathbf{x}| = |\mathbf{y}| = n$ , and each label  $y_i$  has  $s$  different possible discrete values. This model is analogous to maximum entropy models for structured outputs, where expectations can be efficiently calculated by dynamic programming. For a linear-chain CRF of Markov order one:

$$p_\theta(\mathbf{y} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \left( \sum_k \theta_k F_k(\mathbf{x}, \mathbf{y}) \right),$$

where  $F_k(\mathbf{x}, \mathbf{y}) = \sum_i f_k(\mathbf{x}, y_i, y_{i+1}, i)$ , and the partition function  $Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp(\sum_k \theta_k F_k(\mathbf{x}, \mathbf{y}))$ . Given training data  $D = \langle (\mathbf{x}^{(1)}, \mathbf{y}^{(1)})..(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) \rangle$ , the model is traditionally trained by maximizing the log-likelihood  $\mathcal{O}(\theta; D) = \sum_d \log p_\theta(\mathbf{y}^{(d)} | \mathbf{x}^{(d)})$  by gradient ascent where the gradient of the likelihood is:

$$\begin{aligned} \frac{\partial}{\partial \theta_k} \mathcal{O}(\theta; D) &= \sum_d F_k(\mathbf{x}^{(d)}, \mathbf{y}^{(d)}) \\ &\quad - \sum_d \sum_{\mathbf{y}} p_\theta(\mathbf{y} | \mathbf{x}^{(d)}) F_k(\mathbf{x}^{(d)}, \mathbf{y}). \end{aligned}$$

The second term (the expected counts of the features given the model) can be computed in a tractable amount of time, since according to the Markov as-

<sup>2</sup>Often these are more complicated than picking informative features as proposed in this paper. One example of the kind of operator used is the transposition operator proposed by Smith and Eisner (2005).

sumption, the feature expectations can be rewritten:

$$\sum_{\mathbf{y}} p_{\theta}(\mathbf{y}|\mathbf{x}) F_k(\mathbf{x}, \mathbf{y}) = \sum_i \sum_{y_i, y_{i+1}} p_{\theta}(y_i, y_{i+1}|\mathbf{x}) f_k(\mathbf{x}, y_i, y_{i+1}, i).$$

A dynamic program (the forward/backward algorithm) then computes in time  $O(ns^2)$  all the needed probabilities  $p_{\theta}(y_i, y_{i+1})$ , where  $n$  is the sequence length, and  $s$  is the number of labels.

#### 4 Generalized Expectation Criteria for Conditional Random Fields

Prior semi-supervised learning methods have augmented a limited amount of fully labeled data with either unlabeled data or with constraints (e.g. features marked with their majority label). GE criteria can use more information than these previous methods. In particular GE criteria can take advantage of conditional probability distributions of labels given a feature ( $p(y|f_k(x) = 1)$ ). This information provides richer constraints to the model while remaining easily interpretable. People have good intuitions about the relative predictive strength of different features. For example, it is clear that the probability of label PERSON given the feature WORD=JOHN is high, perhaps around 0.95, where as for WORD=BROWN it would be lower, perhaps 0.4. These distributions need not be not estimated with great precision—it is far better to have the freedom to express shades of gray than to be force into a binary supervision signal. Another advantage of using conditional probability distributions as probabilistic constraints is that they can be easily estimated from data. For the feature INITIAL-CAPITAL, we identify all tokens with the feature, and then count the labels with which the feature co-occurs.

GE criteria attempt to match these conditional probability distributions by model expectations on unlabeled data, encouraging, for example, the model to predict that the proportion of the label PERSON given the word “john” should be .95 over all of the unlabeled data.

In general, a GE (generalized expectation) criterion (McCallum et al., 2007) expresses a preference on the value of a model expectation. One kind of preference may be expressed by a distance function

$\Delta$ , a target expectation  $\hat{f}$ , data  $D$ , a function  $f$ , and a model distribution  $p_{\theta}$ , the GE criterion objective function term is  $\Delta(\hat{f}, E[f(x)])$ . For the purposes of this paper, we set the functions to be conditional probability distributions and set  $\Delta(p, q) = D(p||q)$ , the KL-divergence between two distributions.<sup>3</sup> For semi-supervised training of CRFs, we augment the objective function with the regularization term:

$$\mathcal{O}(\theta; D, U) = \sum_d \log p_{\theta}(\mathbf{y}^{(d)}|\mathbf{x}^{(d)}) - \frac{\sum_k \theta_k}{2\sigma^2} - \lambda D(\hat{p}||\tilde{p}_{\theta}),$$

where  $\hat{p}$  is given as a target distribution and

$$\begin{aligned} \tilde{p}_{\theta} &= \tilde{p}_{\theta}(y_j|f_m(\mathbf{x}, j) = 1) \\ &= \frac{1}{U_m} \sum_{\mathbf{x} \in U_m} \sum_{j^*} p_{\theta}(y_{j^*}^*|\mathbf{x}), \end{aligned}$$

with the unnormalized potential

$$\tilde{q}_{\theta} = \tilde{q}_{\theta}(y_j|f_m(\mathbf{x}, j) = 1) = \sum_{\mathbf{x} \in U_m} \sum_{j^*} p_{\theta}(y_{j^*}^*|\mathbf{x}),$$

where  $f_m(\mathbf{x}, j)$  is a feature that depends only on the observation sequence  $\mathbf{x}$ , and  $j^*$  is defined as  $\{j : f_m(\mathbf{x}, j) = 1\}$ , and  $U_m$  is the set of sequences where  $f_m(\mathbf{x}, j)$  is present for some  $j$ .<sup>4</sup>

#### Computing the Gradient

To compute the gradient of the GE criteria,  $D(\hat{p}||\tilde{p}_{\theta})$ , first we drop terms that are constant with respect to the partial derivative, and we derive the gradient as follows:

$$\begin{aligned} \frac{\partial}{\partial \theta_k} \sum_l \hat{p} \log \tilde{q}_{\theta} &= \sum_l \frac{\hat{p}}{\tilde{q}_{\theta}} \frac{\partial}{\partial \theta_k} \tilde{q}_{\theta} \\ &= \sum_l \frac{\hat{p}}{\tilde{q}_{\theta}} \sum_{\mathbf{x} \in U} \sum_{j^*} \frac{\partial}{\partial \theta_k} p_{\theta}(y_{j^*} = l|\mathbf{x}) \\ &= \sum_l \frac{\hat{p}}{\tilde{q}_{\theta}} \sum_{\mathbf{x} \in U} \sum_{j^*} \sum_{y_{-j^*}} \frac{\partial}{\partial \theta_k} p_{\theta}(y_{j^*} = l, \mathbf{y}_{-j^*}|\mathbf{x}), \end{aligned}$$

where  $\mathbf{y}_{-j} = \langle \mathbf{y}_{1..(j-1)} \mathbf{y}_{(j+1)..n} \rangle$ . The last step follows from the definition of the marginal probability

<sup>3</sup>We are actively investigating different choices of distance functions which may have different generalization properties.

<sup>4</sup>This formulation assumes binary features.

$P(y_j|\mathbf{x})$ . Now that we have a familiar form in which we are taking the gradient of a particular label sequence, we can continue:

$$\begin{aligned}
&= \sum_l \frac{\hat{p}}{\tilde{q}_\theta} \sum_{\mathbf{x} \in \mathbf{U}} \sum_{j^*} \sum_{\mathbf{y}_{-j^*}} p_\theta(y_{j^*} = l, \mathbf{y}_{-j^*} | \mathbf{x}) F_k(\mathbf{x}, \mathbf{y}) \\
&\quad - \sum_l \frac{\hat{p}}{\tilde{q}_\theta} \sum_{\mathbf{x} \in \mathbf{U}} \sum_{j^*} \sum_{\mathbf{y}_{-j^*}} p_\theta(y_{j^*} = l, \mathbf{y}_{-j^*} | \mathbf{x}) \\
&\quad \quad \sum_{\mathbf{y}'} p_\theta(\mathbf{y}' | \mathbf{x}) F_k(\mathbf{x}, \mathbf{y}') \\
&= \sum_l \frac{\hat{p}}{\tilde{q}_\theta} \sum_{\mathbf{x} \in \mathbf{U}} \sum_i \sum_{y_i, y_{i+1}} f_k(\mathbf{x}, y_i, y_{i+1}, i) \\
&\quad \quad \sum_{j^*} p_\theta(y_i, y_{i+1}, y_{j^*} = l | \mathbf{x}) \\
&\quad - \sum_l \frac{\hat{p}}{\tilde{q}_\theta} \sum_{\mathbf{x} \in \mathbf{U}} \sum_i \sum_{y_i, y_{i+1}} f_k(\mathbf{x}, y_i, y_{i+1}, i) \\
&\quad \quad p_\theta(y_i, y_{i+1} | \mathbf{x}) \sum_{j^*} p_\theta(y_{j^*} = l | \mathbf{x}).
\end{aligned}$$

After combining terms and rearranging we arrive at the final form of the gradient:

$$\begin{aligned}
&= \sum_{\mathbf{x} \in \mathbf{U}} \sum_i \sum_{y_i, y_{i+1}} f_k(\mathbf{x}, y_i, y_{i+1}, i) \sum_l \frac{\hat{p}}{\tilde{q}_\theta} \times \\
&\quad \left( \sum_{j^*} p_\theta(y_i, y_{i+1}, y_{j^*} = l | \mathbf{x}) - \right. \\
&\quad \quad \left. p_\theta(y_i, y_{i+1} | \mathbf{x}) \sum_{j^*} p_\theta(y_{j^*} = l | \mathbf{x}) \right).
\end{aligned}$$

Here, the second term is easily gathered from forward/backward, but obtaining the first term is somewhat more complicated. Computing this term naively would require multiple runs of constrained forward/backward. Here we present a more efficient method that requires only one run of forward/backward.<sup>5</sup> First we decompose the probability into two parts:  $\sum_{j^*} p_\theta(y_i, y_{i+1}, y_{j^*} = l | \mathbf{x}) = \sum_{j=1}^i p_\theta(y_i, y_{i+1}, y_j = l | \mathbf{x}) I(j \in j^*) + \sum_{j=i+1}^J p_\theta(y_i, y_{i+1}, y_j = l | \mathbf{x}) I(j \in j^*)$ . Next, we show how to compute these terms efficiently. Similar to forward/backward, we build a lattice of intermediate results that then can be used to calculate the

<sup>5</sup>(Kakade et al., 2002) propose a related method that computes  $p(y_{1..i} = l_{1..i} | y_{i+1} = l)$ .

quantity of interest:

$$\begin{aligned}
&\sum_{j=1}^i p_\theta(y_i, y_{i+1}, y_j = l | \mathbf{x}) I(j \in j^*) \\
&= p(y_i, y_{i+1} | \mathbf{x}) \delta(y_i, l) I(i \in j^*) \\
&\quad + \sum_{j=1}^{i-1} p_\theta(y_i, y_{i+1}, y_j = l | \mathbf{x}) I(j \in j^*) \\
&= p(y_i, y_{i+1} | \mathbf{x}) \delta(y_i, l) I(i \in j^*) \\
&\quad + \left( \sum_{y_{i-1}} \sum_{j=1}^{i-1} p_\theta(y_{i-1}, y_i, y_j = l | \mathbf{x}) I(j \in j^*) \right) \\
&\quad \quad p_\theta(y_{i+1} | y_i, \mathbf{x}).
\end{aligned}$$

For efficiency,  $\sum_{y_{i-1}} \sum_{j=1}^{i-1} p_\theta(y_{i-1}, y_i, y_j = l | \mathbf{x}) I(j \in j^*)$  is saved at each stage in the lattice.  $\sum_{j=i+1}^J p_\theta(y_{i-1}, y_i, y_j = l | \mathbf{x}) I(j \in j^*)$  can be computed in the same fashion. To compute the lattices it takes time  $O(ns^2)$ , and one lattice must be computed for each label so the total time is  $O(ns^3)$ .

## 5 Experimental Results

We use the CLASSIFIEDS data provided by Grenager et al. (2005) and compare with results reported by HK06 (Haghighi and Klein, 2006) and CRR07 (Chang et al., 2007). HK06 introduced a set of 33 features along with their majority labels, these are the primary set of additional constraints (Table 1). As HK06 notes, these features are selected using statistics of the labeled data, and here we used similar features here in order to compare with previous results. Though in practice we have found that feature selection is often intuitive, recent work has experimented with automatic feature selection using LDA (Druck et al., 2008). For some of the experiments we also use two sets of 33 additional features that we chose by the same method as HK06, the first 33 of which are also shown in Table 1. We use the same tokenization of the dataset as HK06, and training/test/unsupervised sets of 100 instances each. This data differs slightly from the tokenization used by CRR07. In particular it lacks the newline breaks which might be a useful piece of information.

There are three types of supervised/semi-supervised data used in the experiments. **Labeled instances** are the traditional or conventionally

Label	HK06: 33 Features	33 Added Features
CONTACT	*phone* call *time	please appointment more
FEATURES	kitchen laundry parking	room new large
ROOMMATES	roommate respectful drama	i bit mean
RESTRICTIONS	pets smoking dog	no sorry cats
UTILITIES	utilities pays electricity	water garbage included
AVAILABLE	immediately begin cheaper	*month* now *ordinal*0
SIZE	*number*1*1 br sq	*number*0*1 bedroom bath
PHOTOS	pictures image link	*url*long click photos
RENT	*number*15*1 \$ month	deposit lease rent
NEIGHBORHOOD	close near shopping	located bart downtown
ADDRESS	address carlmont	ave san *ordinal*5 #

Table 1: Features and their associated majority label. Features for each label were chosen by the method described in HK06 – top frequency for that label and not higher frequency for any other label.

		+ SVD features
HK06	53.7%	<b>71.5%</b>
CRF + GE/Heuristic	<b>66.9%</b>	68.3%

Table 2: Accuracy of semi-supervised learning methods with majority labeled features alone. GE outperforms HK06 when neither model has access to SVD features. When SVD features are included, HK06 has an edge in accuracy.

labeled instances used for estimation in traditional CRF training. **Majority labeled features** are features annotated with their majority label.<sup>6</sup> **Labeled features** are features  $m$  where the distribution  $p(y_i|f_m(\mathbf{x}, i))$  has been specified. In Section 5.3 we estimate these distributions from isolated labeled tokens.

We evaluate the system in two scenarios: (1) with feature constraints alone and (2) feature constraints in conjunction with a minimal amount of labeled instances. There is little prior work that demonstrates the use of both scenarios; CRR07 can only be applied when there is some labeled data, while HK06 could be applied in both scenarios though there are no such published experiments.

## 5.1 Majority Labeled Features Only

When using majority labeled features alone, it can be seen in Table 2 that GE is the best performing method. This is important, as it demonstrates that GE out of the box can be used effectively, without tuning and extra modifications.

<sup>6</sup>While HK06 and CRR07 require only majority labeled features, GE criteria use conditional probability distributions of labels given features, and so in order to apply GE we must decide on a particular distribution for each feature constraint. In sections 5.1 and 5.2 we use a simple heuristic to derive distributions from majority label information: we assign .99 probability to the majority label of the feature and divide the remaining probability uniformly among the remainder of the labels.

	Labeled Instances		
	10	25	100
supervised HMM	61.6%	70.0%	76.3%
supervised CRF	64.6%	72.9%	79.4%
CRF+ Entropy Reg.	67.3%	73.7%	79.5%
CRR07	70.9%	74.8%	78.6%
+ inference constraints	<b>74.7%</b>	<b>78.5%</b>	<b>81.7%</b>
CRF+GE/Heuristic	72.6%	76.3%	80.1%

Table 3: Accuracy of semi-supervised learning methods with constraints and limited amounts of training data. Even though CRR07 uses more constraints and requires additional development data for estimating mixture weights, GE still outperforms CRR07 when that system is run without applying constraints during inference. When these constraints are applied during test-time inference, CRR07 has an edge over the CRF trained with GE criteria.

In their original work, HK06 propose a method for generating additional features given a set of “prototype” features (the feature constraints in Table 1), which they demonstrate to be highly effective. In their method, they collect contexts around all words in the corpus, then perform a SVD decomposition. They take the first 50 singular values for all words, and then if a word is within a thresholded distance to a prototype feature, they assign that word a new feature which indicates close similarity to a prototype feature. When SVD features such as these are made available to the systems, HK06 has a higher accuracy.<sup>7</sup> For the remainder of the experiments we use the SVD feature enhanced data sets.<sup>8</sup>

We ran additional experiments with expected gradient methods but found them to be ineffective, reaching around 50% accuracy on the experiments with the additional SVD features, around 20% less than the competing methods.

## 5.2 Majority Labeled Features and Labeled Instances

Labeled instances are available, the technique described in CRR07 can be used. While CRR07 is run on the same data set as used by HK06, a direct comparison is problematic. First, they use additional constraints beyond those used in this paper and those

<sup>7</sup>We generated our own set of SVD features, so they might not match exactly the SVD features described in HK06.

<sup>8</sup>One further experiment HK06 performs which we do not duplicate here is post-processing the label assignments to better handle field boundaries. With this addition they realize another 2.5% improvement.

used by HK06 (e.g. each contiguous label sequence must be at least 3 labels long)—so their results cannot be directly compared. Second, they require additional training data to estimate weights for their soft constraints, and do not measure how much of this additional data is needed. Third, they use a slightly different tokenization procedure. Fourth, CRR07 uses different subsets of labeled training instances than used here. For these reasons, the comparison between the method presented here and CRR07 cannot be exact.

The technique described in CRR07 can be applied in two ways: constraints can be applied during learning, and they can also be applied during inference. We present comparisons with both of these systems in Table 3. CRFs trained with GE criteria consistently outperform CRR07 when no constraints are applied during inference time, even though CRR07 has additional constraints. When the method in CRR07 is applied with constraints in inference time, it is able to outperform CRFs trained with GE. We tried adding the additional constraints described in CRR07 during test-time inference in our system, but found no accuracy improvement. After doing error inspection, those additional constraints weren’t frequently violated by the GE trained method, which also suggests that adding them wouldn’t have a significant effect during training either. It is possible that for GE training there are alternative inference-time constraints that would improve performance, but we didn’t pursue this line of investigation as there are benefits to operating within a formal probabilistic model, and eschewing constraints applied during inference time. Without these constraints, probabilistic models can be combined easily with one another in order to arrive at a joint model, and adding in these constraints at inference time complicates the nature of the combination.

### 5.3 Labeled Features vs. Labeled Instances

In the previous section, the supervision signal was the majority label of each feature.<sup>9</sup> Given a feature of interest, a human can gather a set of tokens that have this feature and label them to discover the cor-

<sup>9</sup>It is not clear how these features would be tagged with majority label in a real use case. Tagging data to discover the majority label could potentially require a large number of tagged instances before the majority label was definitively identified.

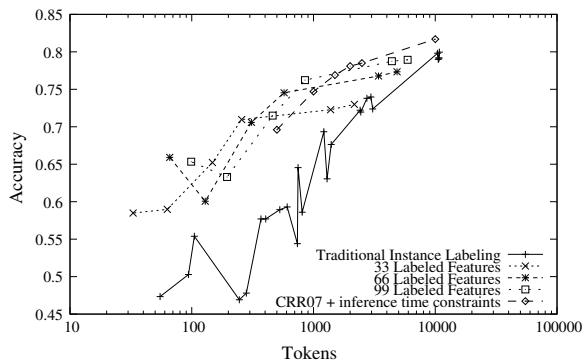


Figure 2: Accuracy of supervised and semi-supervised learning methods for fixed numbers of labeled tokens. Training a GE model with only labeled features significantly outperforms traditional log-likelihood training with labeled instances for comparable numbers of labeled tokens. When training on less than 1500 annotated tokens, it also outperforms CRR07 + inference time constraints, which uses not only labeled tokens but additional constraints and development data for estimating mixture weights.

	Labeled Instances			
	0	10	25	100
HK06	71.5%	-	-	-
GE/Heuristic	68.3%	72.6%	76.3%	80.1%
GE/Sampled	<b>73.0%</b>	<b>74.6%</b>	<b>77.2%</b>	<b>80.5%</b>

Table 4: Accuracy of semi-supervised learning methods comparing the effects of (1) a heuristic for setting conditional distributions of labels given features and (2) estimating this distributions via human annotation. When GE is given feature distributions are better than the simple heuristic it is able to realize considerable gains.

relation between the feature and the labels.<sup>10</sup> While the resulting label distribution information could not be fully utilized by previous methods (HK06 and CRR07 use only the majority label of the word), it can, however, be integrated into the GE criteria by using the distribution from the relative proportions of labels rather than a the previous heuristic distribution. We present a series of experiments that test the advantages of this annotation paradigm.

To simulate a human labeler, we randomly sample (without replacement) tokens with the particular feature in question, and generate a label using the human annotations provided in the data. Then we normalize and smooth the raw counts to obtain a

<sup>10</sup>In this paper we observe a 10x speed-up by using isolated labeled tokens instead of a wholly labeled instances—so even if it takes slightly longer to label isolated tokens, there will still be a substantial gain.



conditional probability distribution over labels given feature. We experiment with samples of 1, 2, 5, 10, 100 tokens per feature, as well as with all available labeled data. We sample instances for labeling exclusively from the training and development data, not from the testing data. We train a model using GE with these estimated conditional probability distributions and compare them with corresponding numbers of tokens of traditionally labeled instances.

Training from labeled features significantly outperforms training from traditional labeled instances for equivalent numbers of labeled tokens (Figure 2). With 1000 labeled tokens, instance-labeling achieves accuracy around 65%, while labeling 33 features reaches 72% accuracy.<sup>11</sup> To achieve the same level of performance as traditional instance-labeling, it can require as much as a factor of ten-fold fewer annotations of feature occurrences. For example, the accuracy achieved after labeling 257 tokens of 33 features is 71% – the same accuracy achieved only after labeling more than 2000 tokens in traditional instance-labeling.<sup>12</sup>

Assuming that labeling one token in isolation takes the same time as labeling one token in a sequence, these results strongly support a new paradigm of labeling in which instead of annotating entire sentences, the human instead selects some key features of interest and labels tokens that have this feature. Particularly intriguing is the flexibility our scenario provides for the selection of “features of interest” to be driven by error analysis.

Table 4 compares the heuristic method described above against sampled conditional probability distributions of labels given features<sup>13</sup>. Sampled distributions yield consistent improvements over the heuristic method. The accuracy with no labeled instances (73.0%) is better than HK06 (71.5%), which demonstrates that the precisely estimated feature distributions are helpful for improving accuracy.

Though accuracy begins to level off with distri-

<sup>11</sup>Labeling 99 features with 1000 tokens reaches nearly 76%.

<sup>12</sup>Accuracy at one labeled token per feature is much worse than accuracy with majority label information. This due to the noise introduced by sampling, as there is the potential for a relatively rare label be sampled and labeled, and thereby train the system on a non-canonical supervision signal.

<sup>13</sup>Where the tokens labeled is the total available number in the data, roughly 2500 tokens.

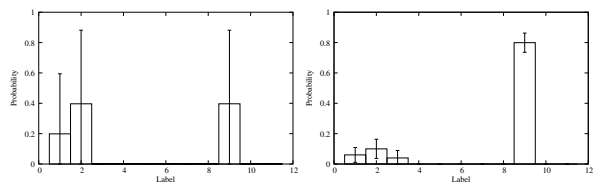


Figure 3: From left to right: distributions (with standard error) for the feature WORD=ADDRESS obtained from sampling, using 1 sample per feature and 10 samples per feature. Labels 1, 2, 3, and 9 are (respectively) FEATURES, CONTACT, SIZE, and ADDRESS. Instead of more precisely estimating these distributions, it is more beneficial to label a larger set of features.

butions over the original set of 33 labeled features, we ran additional experiments with 66 and 99 labeled features, whose results are also shown in Figure 2.<sup>14</sup> The graph shows that with an increased number of labeled features, for the same numbers of labeled tokens, accuracy can be improved. The reason behind this is clear—while there is some gain from increased precision of probability estimates (as they asymptotically approach their “true” values as shown in Figure 3), there is more information to be gained from rougher estimates of a larger set of features. One final point about these additional features is that their distributions are less peaked than the original feature set. Where the original feature set distribution has entropy of 8.8, the first 33 added features have an entropy of 22.95. Surprisingly, even *ambiguous* feature constraints are able to improve accuracy.

## 6 Conclusion

We have presented generalized expectation criteria for linear-chain conditional random fields, a new semi-supervised training method that makes use of labeled features rather than labeled instances. Previous semi-supervised methods have typically used ad-hoc feature majority label assignments as constraints. Our new method uses conditional probability distributions of labels given features and can dramatically reduce annotation time. When these distributions are estimated by means of annotated feature occurrences in context, there is as much as a ten-fold reduction in the annotation time that is required in order to achieve the same level of accuracy over traditional instance-labeling.

<sup>14</sup>Also note that for less than 1500 tokens of labeling, the 99 labeled features outperform CRR07 with inference time constraints.

## References

- R. K. Ando and T. Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6.
- M.-W. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*.
- G. Druck, G. Mann, and A. McCallum. 2007. Leveraging existing resources using generalized expectation criteria. In *NIPS Workshop on Learning Problem Design*.
- G. Druck, G. S. Mann, and A. McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *SIGIR*.
- D. Freitag. 2004. Trained named entity recognition using distributional clusters. In *EMNLP*.
- Y. Grandvalet and Y. Bengio. 2004. Semi-supervised learning by entropy minimization. In *NIPS*.
- T. Grenager, D. Klein, and C. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *ACL*.
- A. Haghighi and D. Klein. 2006. Prototype-driver learning for sequence models. In *NAACL*.
- F. Jiao, S. Wang, C.-H. Lee, R. Greiner, and D. Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *COLING/ACL*.
- Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *ICML*.
- S. Kakade, Y.-W. Teh, and S. Roweis. 2002. An alternate objective function for markovian fields. In *ICML*.
- G. Mann and A. McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *ICML*.
- A. McCallum, G. S. Mann, and G. Druck. 2007. Generalized expectation criteria. Computer science technical note, University of Massachusetts, Amherst, MA.
- S. Miller, J. Guinness, and A. Zamanian. 2004. Name tagging with word clusters and discriminative training. In *ACL*.
- K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 1998. Learning to classify text from labeled and unlabeled documents. In *AAAI*.
- A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. 2007. Hidden-state conditional random fields. In *PAMI*.
- H. Raghavan, O. Madani, and R. Jones. 2006. Active learning with feedback on both features and instances. *JMLR*.
- R. Salakhutdinov, S. Roweis, and Z. Ghahramani. 2003. Optimization with em and expectation-conjugate-gradient. In *ICML*.
- R. Schapire, M. Rochery, M. Rahim, and N. Gupta. 2002. Incorporating prior knowledge into boosting. In *ICML*.
- N. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL*.
- Martin Szummer and Tommi Jaakkola. 2002. Partially labeled classification with markov random walks. In *NIPS*, volume 14.
- X. Zhu and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, CMU.
- X. Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison. [http://www.cs.wisc.edu/~jerryzhu/pub/ssl\\_survey.pdf](http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf).

# Analyzing the Errors of Unsupervised Learning

Percy Liang     Dan Klein

Computer Science Division, EECS Department  
University of California at Berkeley  
Berkeley, CA 94720  
{pliang,klein}@cs.berkeley.edu

## Abstract

We identify four types of errors that unsupervised induction systems make and study each one in turn. Our contributions include (1) using a *meta-model* to analyze the incorrect biases of a model in a systematic way, (2) providing an efficient and robust method of measuring distance between two parameter settings of a model, and (3) showing that local optima issues which typically plague EM can be somewhat alleviated by increasing the number of training examples. We conduct our analyses on three models: the HMM, the PCFG, and a simple dependency model.

## 1 Introduction

The unsupervised induction of linguistic structure from raw text is an important problem both for understanding language acquisition and for building language processing systems such as parsers from limited resources. Early work on inducing grammars via EM encountered two serious obstacles: the inappropriateness of the likelihood objective and the tendency of EM to get stuck in local optima. Without additional constraints on bracketing (Pereira and Shabes, 1992) or on allowable rewrite rules (Carroll and Charniak, 1992), unsupervised grammar learning was ineffective.

Since then, there has been a large body of work addressing the flaws of the EM-based approach. Syntactic models empirically more learnable than PCFGs have been developed (Clark, 2001; Klein and Manning, 2004). Smith and Eisner (2005) proposed a new objective function; Smith and Eisner (2006) introduced a new training procedure. Bayesian approaches can also improve performance (Goldwater and Griffiths, 2007; Johnson, 2007; Kurihara and Sato, 2006).

Though these methods have improved induction accuracy, at the core they all still involve optimizing non-convex objective functions related to the likelihood of some model, and thus are not completely immune to the difficulties associated with early approaches. It is therefore important to better understand the behavior of unsupervised induction systems in general.

In this paper, we take a step back and present a more statistical view of unsupervised learning in the context of grammar induction. We identify four types of error that a system can make: *approximation*, *identifiability*, *estimation*, and *optimization* errors (see Figure 1). We try to isolate each one in turn and study its properties.

Approximation error is caused by a mis-match between the likelihood objective optimized by EM and the true relationship between sentences and their syntactic structures. Our key idea for understanding this mis-match is to “cheat” and initialize EM with the true relationship and then study the ways in which EM repurposes our desired syntactic structures to increase likelihood. We present a *meta-model* of the changes that EM makes and show how this tool can shed some light on the undesired biases of the HMM, the PCFG, and the dependency model with valence (Klein and Manning, 2004).

Identifiability error can be incurred when two distinct parameter settings yield the same probability distribution over sentences. One type of non-identifiability present in HMMs and PCFGs is label symmetry, which even makes computing a meaningful distance between parameters NP-hard. We present a method to obtain lower and upper bounds on such a distance.

Estimation error arises from having too few training examples, and optimization error stems from

EM getting stuck in local optima. While it is to be expected that estimation error should decrease as the amount of data increases, we show that optimization error can also decrease. We present striking experiments showing that if our data actually comes from the model family we are learning with, we can sometimes recover the true parameters by simply running EM without clever initialization. This result runs counter to the conventional attitude that EM is doomed to local optima; it suggests that increasing the amount of data might be an effective way to partially combat local optima.

## 2 Unsupervised models

Let  $\mathbf{x}$  denote an input sentence and  $\mathbf{y}$  denote the unobserved desired output (e.g., a parse tree). We consider a model family  $\mathcal{P} = \{p_\theta(\mathbf{x}, \mathbf{y}) : \theta \in \Theta\}$ . For example, if  $\mathcal{P}$  is the set of all PCFGs, then the parameters  $\theta$  would specify all the rule probabilities of a particular grammar. We sometimes use  $\theta$  and  $p_\theta$  interchangeably to simplify notation. In this paper, we analyze the following three model families:

In the **HMM**, the input  $\mathbf{x}$  is a sequence of words and the output  $\mathbf{y}$  is the corresponding sequence of part-of-speech tags.

In the **PCFG**, the input  $\mathbf{x}$  is a sequence of POS tags and the output  $\mathbf{y}$  is a binary parse tree with yield  $\mathbf{x}$ . We represent  $\mathbf{y}$  as a multiset of binary rewrites of the form  $(y \rightarrow y_1 y_2)$ , where  $y$  is a nonterminal and  $y_1, y_2$  can be either nonterminals or terminals.

In the dependency model with valence (**DMV**) (Klein and Manning, 2004), the input  $\mathbf{x} = (x_1, \dots, x_m)$  is a sequence of POS tags and the output  $\mathbf{y}$  specifies the directed links of a projective dependency tree. The generative model is as follows: for each head  $x_i$ , we generate an independent sequence of arguments to the left and to the right from a direction-dependent distribution over tags. At each point, we stop with a probability parametrized by the direction and whether any arguments have already been generated in that direction. See Klein and Manning (2004) for a formal description.

In all our experiments, we used the Wall Street Journal (WSJ) portion of the Penn Treebank. We binarized the PCFG trees and created gold dependency trees according to the Collins head rules. We trained 45-state HMMs on all 49208 sentences, 11-state

PCFGs on WSJ-10 (7424 sentences) and DMVs on WSJ-20 (25523 sentences) (Klein and Manning, 2004). We ran EM for 100 iterations with the parameters initialized uniformly (always plus a small amount of random noise). We evaluated the HMM and PCFG by mapping model states to Treebank tags to maximize accuracy.

## 3 Decomposition of errors

Now we will describe the four types of errors (Figure 1) more formally. Let  $p^*(\mathbf{x}, \mathbf{y})$  denote the distribution which governs the true relationship between the input  $\mathbf{x}$  and output  $\mathbf{y}$ . In general,  $p^*$  does not live in our model family  $\mathcal{P}$ . We are presented with a set of  $n$  unlabeled examples  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}$  drawn i.i.d. from the true  $p^*$ . In unsupervised induction, our goal is to approximate  $p^*$  by some model  $p_\theta \in \mathcal{P}$  in terms of strong generative capacity. A standard approach is to use the EM algorithm to optimize the empirical likelihood  $\hat{\mathbb{E}} \log p_\theta(\mathbf{x})$ .<sup>1</sup> However, EM only finds a local maximum, which we denote  $\hat{\theta}_{\text{EM}}$ , so there is a *discrepancy* between what we get ( $p_{\hat{\theta}_{\text{EM}}}$ ) and what we want ( $p^*$ ).

We will define this discrepancy later, but for now, it suffices to remark that the discrepancy depends on the distribution over  $\mathbf{y}$  whereas learning depends only on the distribution over  $\mathbf{x}$ . This is an important property that distinguishes unsupervised induction from more standard supervised learning or density estimation scenarios.

Now let us walk through the four types of error bottom up. First,  $\hat{\theta}_{\text{EM}}$ , the local maximum found by EM, is in general different from  $\hat{\theta} \in \operatorname{argmax}_\theta \hat{\mathbb{E}} \log p_\theta(\mathbf{x})$ , any global maximum, which we could find given unlimited computational resources. *Optimization error* refers to the discrepancy between  $\hat{\theta}$  and  $\hat{\theta}_{\text{EM}}$ .

Second, our training data is only a noisy sample from the true  $p^*$ . If we had infinite data, we would choose an optimal parameter setting under the model,  $\theta_2^* \in \operatorname{argmax}_\theta \mathbb{E} \log p_\theta(\mathbf{x})$ , where now the expectation  $\mathbb{E}$  is taken with respect to the true  $p^*$  instead of the training data. The discrepancy between  $\theta_2^*$  and  $\hat{\theta}$  is the *estimation error*.

Note that  $\theta_2^*$  might not be unique. Let  $\theta_1^*$  denote

<sup>1</sup>Here, the expectation  $\hat{\mathbb{E}} f(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f(\mathbf{x}^{(i)})$  denotes averaging some function  $f$  over the training data.

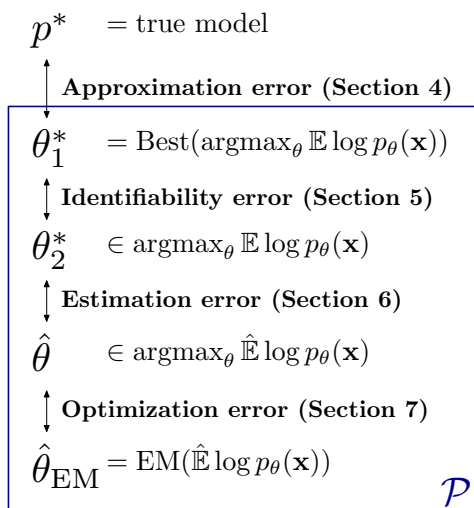


Figure 1: The discrepancy between what we get ( $\hat{\theta}_{\text{EM}}$ ) and what we want ( $p^*$ ) can be decomposed into four types of errors. The box represents our model family  $\mathcal{P}$ , which is the set of possible parametrized distributions we can represent. Best( $S$ ) returns the  $\theta \in S$  which has the smallest discrepancy with  $p^*$ .

the maximizer of  $\mathbb{E} \log p_\theta(\mathbf{x})$  that has the smallest discrepancy with  $p^*$ . Since  $\theta_1^*$  and  $\theta_2^*$  have the same value under the objective function, we would not be able to choose  $\theta_1^*$  over  $\theta_2^*$ , even with infinite data or unlimited computation. Identifiability error refers to the discrepancy between  $\theta_1^*$  and  $\theta_2^*$ .

Finally, the model family  $\mathcal{P}$  has fundamental limitations. *Approximation error* refers to the discrepancy between  $p^*$  and  $p_{\theta_1^*}$ . Note that  $\theta_1^*$  is not necessarily the best in  $\mathcal{P}$ . If we had labeled data, we could find a parameter setting in  $\mathcal{P}$  which is closer to  $p^*$  by optimizing joint likelihood  $\mathbb{E} \log p_\theta(\mathbf{x}, \mathbf{y})$  (generative training) or even conditional likelihood  $\mathbb{E} \log p_\theta(\mathbf{y} | \mathbf{x})$  (discriminative training).

In the remaining sections, we try to study each of the four errors in isolation. In practice, since it is difficult to work with some of the parameter settings that participate in the error decomposition, we use computationally feasible surrogates so that the error under study remains the dominant effect.

## 4 Approximation error

We start by analyzing approximation error, the discrepancy between  $p^*$  and  $p_{\theta_1^*}$  (the model found by optimizing likelihood), a point which has been dis-

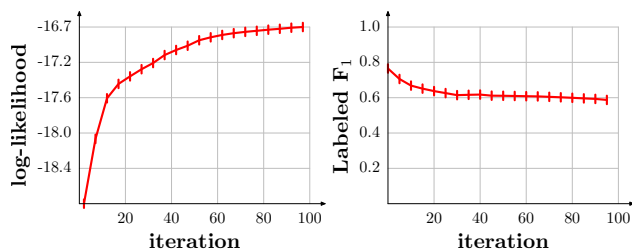


Figure 2: For the PCFG, when we initialize EM with the supervised estimate  $\hat{\theta}_{\text{gen}}$ , the likelihood increases but the accuracy decreases.

cussed by many authors (Merialdo, 1994; Smith and Eisner, 2005; Haghghi and Klein, 2006).<sup>2</sup>

To confront the question of specifically how the likelihood diverges from prediction accuracy, we perform the following experiment: we initialize EM with the supervised estimate<sup>3</sup>  $\hat{\theta}_{\text{gen}} = \operatorname{argmax}_\theta \hat{\mathbb{E}} \log p_\theta(\mathbf{x}, \mathbf{y})$ , which acts as a surrogate for  $p^*$ . As we run EM, the likelihood increases but the accuracy decreases (Figure 2 shows this trend for the PCFG; the HMM and DMV models behave similarly). We believe that the initial iterations of EM contain valuable information about the incorrect biases of these models. However, EM is changing hundreds of thousands of parameters at once in a non-trivial way, so we need a way of characterizing the important changes.

One broad observation we can make is that the first iteration of EM reinforces the systematic mistakes of the supervised initializer. In the first E-step, the posterior counts that are computed summarize the predictions of the supervised system. If these match the empirical counts, then the M-step does not change the parameters. But if the supervised system predicts too many JJs, for example, then the M-step will update the parameters to reinforce this bias.

### 4.1 A meta-model for analyzing EM

We would like to go further and characterize the specific changes EM makes. An initial approach is to find the parameters that changed the most during the first iteration (weighted by the correspond-

<sup>2</sup>Here, we think of discrepancy between  $p$  and  $p'$  as the error incurred when using  $p'$  for prediction on examples generated from  $p$ ; in symbols,  $\mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim p} \text{loss}(\mathbf{y}, \operatorname{argmax}_{\mathbf{y}'} p'(\mathbf{y}' | \mathbf{x}))$ .

<sup>3</sup>For all our models, the supervised estimate is solved in closed form by taking ratios of counts.

ing expected counts computed in the E-step). For the HMM, the three most changed parameters are the transitions 2:DT→8:JJ, START→0:NNP, and 8:JJ→3:NN.<sup>4</sup> If we delve deeper, we can see that 2:DT→3:NN (the parameter with the 10th largest change) fell and 2:DT→8:JJ rose. After checking with a few examples, we can then deduce that some nouns were retagged as adjectives. Unfortunately, this type of ad-hoc reasoning requires considerable manual effort and is rather subjective.

Instead, we propose using a general *meta-model* to analyze the changes EM makes in an automatic and objective way. Instead of treating parameters as the primary object of study, we look at predictions made by the model and study how they change over time. While a model is a distribution over sentences, a meta-model is a distribution over how the predictions of the model change.

Let  $R(\mathbf{y})$  denote the set of *parts* of a prediction  $\mathbf{y}$  that we are interested in tracking. Each part  $(c, l) \in R(\mathbf{y})$  consists of a *configuration*  $c$  and a *location*  $l$ . For a PCFG, we define a configuration to be a rewrite rule (e.g.,  $c = \text{PP} \rightarrow \text{IN NP}$ ), and a location  $l = [i, k, j]$  to be a span  $[i, j]$  split at  $k$ , where the rewrite  $c$  is applied.

In this work, each configuration is associated with a parameter of the model, but in general, a configuration could be a larger unit such as a subtree, allowing one to track more complex changes. The size of a configuration governs how much the meta-model generalizes from individual examples.

Let  $\mathbf{y}^{(i,t)}$  denote the model prediction on the  $i$ -th training example after  $t$  iterations of EM. To simplify notation, we write  $R_t = R(\mathbf{y}^{(i,t)})$ . The meta-model explains how  $R_t$  became  $R_{t+1}$ .<sup>5</sup>

In general, we expect a part in  $R_{t+1}$  to be explained by a part in  $R_t$  that has a similar location and furthermore, we expect the locations of the two parts to be related in some consistent way. The meta-model uses two notions to formalize this idea: a *distance*  $d(l, l')$  and a *relation*  $r(l, l')$ . For the PCFG,  $d(l, l')$  is the number of positions among  $i, j, k$  that are the same as the corresponding ones in  $l'$ , and  $r((i, k, j), (i', k', j')) = (\text{sign}(i - i'), \text{sign}(j -$

$j'), \text{sign}(k - k'))$  is one of  $3^3$  values. We define a *migration* as a triple  $(c, c', r(l, l'))$ ; this is the unit of change we want to extract from the meta-model.

Our meta-model provides the following generative story of how  $R_t$  becomes  $R_{t+1}$ : each new part  $(c', l') \in R_{t+1}$  chooses an old part  $(c, l) \in R_t$  with some probability that depends on (1) the distance between the locations  $l$  and  $l'$  and (2) the likelihood of the particular migration. Formally:

$$p^{\text{meta}}(R_{t+1} | R_t) = \prod_{(c', l') \in R_{t+1}} \sum_{(c, l) \in R_t} Z_l^{-1} e^{-\alpha d(l, l')} p(c' | c, r(l, l')),$$

where  $Z_l = \sum_{(c, l) \in R_t} e^{-\alpha d(l, l')}$  is a normalization constant, and  $\alpha$  is a hyperparameter controlling the possibility of distant migrations (set to 3 in our experiments).

We learn the parameters of the meta-model with an EM algorithm similar to the one for IBM model 1. Fortunately, the likelihood objective is convex, so we need not worry about local optima.

## 4.2 Results of the meta-model

We used our meta-model to analyze the approximation errors of the HMM, DMV, and PCFG. For these models, we initialized EM with the supervised estimate  $\hat{\theta}_{\text{gen}}$  and collected the model predictions as EM ran. We then trained the meta-model on the predictions between successive iterations. The meta-model gives us an expected count for each migration. Figure 3 lists the migrations with the highest expected counts.

From these migrations, we can see that EM tries to explain  $\mathbf{x}$  better by making the corresponding  $\mathbf{y}$  more *regular*. In fact, many of the HMM migrations on the first iteration attempt to resolve inconsistencies in gold tags. For example, noun adjuncts (e.g., *stock-index*), tagged as both nouns and adjectives in the Treebank, tend to become consolidated under adjectives, as captured by migration (B). EM also re-purposes under-utilized states to better capture distributional similarities. For example, state 24 has migrated to state 40 (N), both of which are now dominated by proper nouns. State 40 initially contained only #, but was quickly overrun with distributionally similar proper nouns such as *Oct.* and *Chap-ter*, which also precede numbers, just as # does.

<sup>4</sup>Here 2:DT means state 2 of the HMM, which was greedily mapped to DT.

<sup>5</sup>If the same part appears in both  $R_t$  and  $R_{t+1}$ , we remove it from both sets.

Iteration 0→1	Iteration 1→2	Iteration 2→3	Iteration 3→4	Iteration 4→5
(A) START → <del>4:NN</del> 24:NNP	(D) <del>4:NN</del> → 4:NN	(G) <del>24:NNP</del> → U.S.	(J) <del>11:RB</del> 32:RP → up	(M) <del>24:NNP</del> 34:\$ → 15:CD
(B) <del>4:NN</del> 8:JJ → 4:NN	(E) START → <del>4:NN</del> 24:NNP	(H) <del>24:NNP</del> → 4:NN	(K) <del>24:NNP</del> 8:JJ → U.S.	(N) 2:IN → <del>24:NNP</del> 40:NNP
(C) 24:NNP → <del>24:NNP</del> 36:NNPS	(F) <del>8:JJ</del> 11:RB → 27:TO	(I) 3:DT → <del>24:NNP</del> 8:JJ	(L) 19:, → <del>11:RB</del> 32:RP	(O) <del>11:RB</del> 32:RP → down

(a) Top HMM migrations. Example: migration (D) means a NN→NN transition is replaced by JJ→NN.

Iteration 0→1	Iteration 1→2	Iteration 2→3	Iteration 3→4	Iteration 4→5
(A) DT NN NN	(D) NNP NNP NNP	(G) DT JJ NNS	(J) DT JJ NN	(M) POS JJ NN
(B) JJ NN NN	(E) NNP NNP NNP	(H) MD RB VB	(K) DT NNP NN	(N) NNS RB VBP
(C) NNP NNP	(F) DT NNP NNP	(I) VBP RB VB	(L) PRP\$ JJ NN	(O) NNS RB VBD

(b) Top DMV migrations. Example: migration (A) means a DT attaches to the closer NN.

Iteration 0→1	Iteration 1→2	Iteration 2→3	Iteration 3→4	Iteration 4→5
(A) RB <sup>4:S</sup> 1:VP RB 1:VP	(D) NNP <sup>0:NP</sup> 0:NP NNP NNP	(G) DT <sup>0:NP</sup> 0:NP DT NN	(J) TO <sup>1:VP</sup> VB TO VB	(M) CD <sup>0:NP</sup> NN CD NN
(B) <sup>0:NP</sup> 2:PP 1:VP 1:VP	(E) <sup>1:VP</sup> 2:PP 1:VP 1:VP	(H) <sup>0:NP</sup> 4:S 1:VP 0:NP 1:VP	(K) MD <sup>1:VP</sup> 1:VP MD VB	(N) VBD <sup>1:VP</sup> 3:ADJP VBD 1:VP
(C) VBZ <sup>1:VP</sup> 0:NP VBZ 1:VP	(F) <sup>0:NP</sup> 4:S 1:VP 0:NP 1:VP	(I) TO <sup>1:VP</sup> VB TO VB	(L) NNP <sup>0:NP</sup> NNP NNP NNP	(O) <sup>0:NP</sup> NN 0:NP NN

(c) Top PCFG migrations. Example: migration (D) means a NP→NNP NP rewrite is replaced by NP→NNP NNP, where the new NNP right child spans less than the old NP right child.

Figure 3: We show the prominent migrations that occur during the first 5 iterations of EM for the HMM, DMV, and PCFG, as recovered by our meta-model. We sort the migrations across each iteration by their expected counts under the meta-model and show the top 3. Iteration 0 corresponds to the correct outputs. Blue indicates the new iteration, red indicates the old.

DMV migrations also try to regularize model predictions, but in a different way—in terms of the number of arguments. Because the stop probability is different for adjacent and non-adjacent arguments, it is statistically much cheaper to generate one argument rather than two or more. For example, if we train a DMV on only DT JJ NN, it can fit the data perfectly by using a chain of single arguments, but perfect fit is not possible if NN generates both DT and JJ (which is the desired structure); this explains migration (J). Indeed, we observed that the variance of the number of arguments decreases with more EM iterations (for NN, from 1.38 to 0.41).

In general, low-entropy conditional distributions are preferred. Migration (H) explains how adverbs now consistently attach to verbs rather than modals. After a few iterations, the modal has committed itself to generating exactly one verb to the right,

which is statistically advantageous because there must be a verb after a modal, while the adverb is optional. This leaves the verb to generate the adverb.

The PCFG migrations regularize categories in a manner similar to the HMM, but with the added complexity of changing bracketing structures. For example, sentential adverbs are re-analyzed as VP adverbs (A). Sometimes, multiple migrations explain the same phenomenon.<sup>6</sup> For example, migrations (B) and (C) indicate that PPs that previously attached to NPs are now raised to the verbal level. Tree rotation is another common phenomenon, leading to many left-branching structures (D,G,H). The migrations that happen during one iteration can also trigger additional migrations in the next. For example, the raising of the PP (B,C) inspires more of the

<sup>6</sup>We could consolidate these migrations by using larger configurations, but at the risk of decreased generalization.

same raising (E). As another example, migration (I) regularizes TO VB infinitival clauses into PPs, and this momentum carries over to the next iteration with even greater force (J).

In summary, the meta-model facilitates our analyses by automatically identifying the broad trends. We believe that the central idea of modeling the errors of a system is a powerful one which can be used to analyze a wide range of models, both supervised and unsupervised.

## 5 Identifiability error

While approximation error is incurred when likelihood diverges from accuracy, identifiability error is concerned with the case where likelihood is indifferent to accuracy.

We say a set of parameters  $S$  is *identifiable* (in terms of  $\mathbf{x}$ ) if  $p_\theta(\mathbf{x}) \neq p_{\theta'}(\mathbf{x})$  for every  $\theta, \theta' \in S$  where  $\theta \neq \theta'$ .<sup>7</sup> In general, identifiability error is incurred when the set of maximizers of  $\mathbb{E} \log p_\theta(\mathbf{x})$  is non-identifiable.<sup>8</sup>

Label symmetry is perhaps the most familiar example of non-identifiability and is intrinsic to models with hidden labels (HMM and PCFG, but not DMV). We can permute the hidden labels without changing the objective function or even the nature of the solution, so there is no reason to prefer one permutation over another. While seemingly benign, this symmetry actually presents a serious challenge in measuring discrepancy (Section 5.1).

Grenager et al. (2005) augments an HMM to allow emission from a generic stopword distribution at any position with probability  $q$ . Their model would definitely not be identifiable if  $q$  were a free parameter, since we can set  $q$  to 0 and just mix in the stopword distribution with each of the other emission distributions to obtain a different parameter setting yielding the same overall distribution. This is a case where our notion of desired structure is absent in the likelihood, and a prior over parameters could help break ties.

<sup>7</sup>For our three model families,  $\theta$  is identifiable in terms of  $(\mathbf{x}, \mathbf{y})$ , but not in terms of  $\mathbf{x}$  alone.

<sup>8</sup>We emphasize that non-identifiability is in terms of  $\mathbf{x}$ , so two parameter settings could still induce the same marginal distribution on  $\mathbf{x}$  (weak generative capacity) while having different joint distributions on  $(\mathbf{x}, \mathbf{y})$  (strong generative capacity). Recall that discrepancy depends on the latter.

The above non-identifiabilities apply to all parameter settings, but another type of non-identifiability concerns only the maximizers of  $\mathbb{E} \log p_\theta(\mathbf{x})$ . Suppose the true data comes from a  $K$ -state HMM. If we attempt to fit an HMM with  $K + 1$  states, we can split any one of the  $K$  states and maintain the same distribution on  $\mathbf{x}$ . Or, if we learn a PCFG on the same HMM data, then we can choose either the left- or right-branching chain structures, which both mimic the true HMM equally well.

### 5.1 Permutation-invariant distance

KL-divergence is a natural measure of discrepancy between two distributions, but it is somewhat non-trivial to compute—for our three recursive models, it requires solving fixed point equations, and becomes completely intractable in face of label symmetry. Thus we propose a more manageable alternative:

$$d_\mu(\theta \parallel \theta') \stackrel{\text{def}}{=} \frac{\sum_j \mu_j |\theta_j - \theta'_j|}{\sum_j \mu_j}, \quad (1)$$

where we weight the difference between the  $j$ -th component of the parameter vectors by  $\mu_j$ , the  $j$ -th expected sufficient statistic with respect to  $p_\theta$  (the expected counts computed in the E-step).<sup>9</sup> Unlike KL, our distance  $d_\mu$  is only defined on distributions in the model family and is not invariant to reparametrization. Like KL,  $d_\mu$  is asymmetric, with the first argument holding the status of being the “true” parameter setting. In our case, the parameters are conditional probabilities, so  $0 \leq d_\mu(\theta \parallel \theta') \leq 1$ , so we can interpret  $d_\mu$  as an expected difference between these probabilities.

Unfortunately, label symmetry can wreak havoc on our distance measure  $d_\mu$ . Suppose we want to measure the distance between  $\theta$  and  $\theta'$ . If  $\theta'$  is simply  $\theta$  with the labels permuted, then  $d_\mu(\theta \parallel \theta')$  would be substantial even though the distance ought to be zero. We define a revised distance to correct for this by taking the minimum distance over all label permutations:

$$D_\mu(\theta \parallel \theta') = \min_\pi d_\mu(\theta \parallel \pi(\theta')), \quad (2)$$

<sup>9</sup>Without this factor, rarely used components could contribute to the sum as much as frequently used ones, thus, making the distance overly pessimistic.



where  $\pi(\theta')$  denotes the parameter setting resulting from permuting the labels according to  $\pi$ . (The DMV has no label symmetries, so just  $d_\mu$  works.)

For mixture models, we can compute  $D_\mu(\theta \parallel \theta')$  efficiently as follows. Note that each term in the summation of (1) is associated with one of the  $K$  labels. We can form a  $K \times K$  matrix  $M$ , where each entry  $M_{ij}$  is the distance between the parameters involving label  $i$  of  $\theta$  and label  $j$  of  $\theta'$ .  $D_\mu(\theta \parallel \theta')$  can then be computed by finding a maximum weighted bipartite matching on  $M$  using the  $O(K^3)$  Hungarian algorithm (Kuhn, 1955).

For models such as the HMM and PCFG, computing  $D_\mu$  is NP-hard, since the summation in  $d_\mu$  (1) contains both first-order terms which depend on one label (e.g., emission parameters) and higher-order terms which depend on more than one label (e.g., transitions or rewrites). We cannot capture these problematic higher-order dependencies in  $M$ .

However, we can bound  $D_\mu(\theta \parallel \theta')$  as follows. We create  $M$  using only first-order terms and find the best matching (permutation) to obtain a lower bound  $\underline{D}_\mu$  and an associated permutation  $\pi_0$  achieving it. Since  $D_\mu(\theta \parallel \theta')$  takes the minimum over all permutations,  $d_\mu(\theta \parallel \pi(\theta'))$  is an upper bound for any  $\pi$ , in particular for  $\pi = \pi_0$ . We then use a local search procedure that changes  $\pi$  to further tighten the upper bound. Let  $\overline{D}_\mu$  denote the final value.

## 6 Estimation error

Thus far, we have considered approximation and identifiability errors, which have to do with flaws of the model. The remaining errors have to do with how well we can fit the model. To focus on these errors, we consider the case where the true model is in our family ( $p^* \in \mathcal{P}$ ). To keep the setting as realistic as possible, we do supervised learning on real labeled data to obtain  $\theta^* = \operatorname{argmax}_\theta \mathbb{E} \log p(\mathbf{x}, \mathbf{y})$ . We then throw away our real data and let  $p^* = p_{\theta^*}$ . Now we start anew: sample new artificial data from  $\theta^*$ , learn a model using this artificial data, and see how close we get to recovering  $\theta^*$ .

In order to compute estimation error, we need to compare  $\theta^*$  with  $\hat{\theta}$ , the global maximizer of the likelihood on our generated data. However, we cannot compute  $\hat{\theta}$  exactly. Let us therefore first consider the simpler supervised scenario. Here,  $\hat{\theta}_{\text{gen}}$  has a closed

form solution, so there is no optimization error. Using our distance  $D_\mu$  (defined in Section 5.1) to quantify estimation error, we see that, for the HMM,  $\hat{\theta}_{\text{gen}}$  quickly approaches  $\theta^*$  as we increase the amount of data (Table 1).

# examples	500	5K	50K	500K
$\underline{D}_\mu(\theta^* \parallel \hat{\theta}_{\text{gen}})$	0.003	6.3e-4	2.7e-4	8.5e-5
$\overline{D}_\mu(\theta^* \parallel \hat{\theta}_{\text{gen}})$	0.005	0.001	5.2e-4	1.7e-4
$\underline{D}_\mu(\theta^* \parallel \hat{\theta}_{\text{gen-EM}})$	0.022	0.018	0.008	0.002
$\overline{D}_\mu(\theta^* \parallel \hat{\theta}_{\text{gen-EM}})$	0.049	0.039	0.016	0.004

Table 1: Lower and upper bounds on the distance from the true  $\theta^*$  for the HMM as we increase the number of examples.

In the unsupervised case, we use the following procedure to obtain a surrogate for  $\hat{\theta}$ : initialize EM with the supervised estimate  $\hat{\theta}_{\text{gen}}$  and run EM for 100 iterations. Let  $\hat{\theta}_{\text{gen-EM}}$  denote the final parameters, which should be representative of  $\hat{\theta}$ . Table 1 shows that the estimation error of  $\hat{\theta}_{\text{gen-EM}}$  is an order of magnitude higher than that of  $\hat{\theta}_{\text{gen}}$ , which is to be expected since  $\hat{\theta}_{\text{gen-EM}}$  does not have access to labeled data. However, this error can also be driven down given a moderate number of examples.

## 7 Optimization error

Finally, we study optimization error, which is the discrepancy between the global maximizer  $\hat{\theta}$  and  $\hat{\theta}_{\text{EM}}$ , the result of running EM starting from a uniform initialization (plus some small noise). As before, we cannot compute  $\hat{\theta}$ , so we use  $\hat{\theta}_{\text{gen-EM}}$  as a surrogate. Also, instead of comparing  $\hat{\theta}_{\text{gen-EM}}$  and  $\hat{\theta}$  with each other, we compare each of their discrepancies with respect to  $\theta^*$ .

Let us first consider optimization error in terms of prediction error. The first observation is that there is a gap between the prediction accuracies of  $\hat{\theta}_{\text{gen-EM}}$  and  $\hat{\theta}_{\text{EM}}$ , but this gap shrinks considerably as we increase the number of examples. Figures 4(a,b,c) support this for all three model families: for the HMM, both  $\hat{\theta}_{\text{gen-EM}}$  and  $\hat{\theta}_{\text{EM}}$  eventually achieve around 90% accuracy; for the DMV, 85%. For the PCFG,  $\hat{\theta}_{\text{EM}}$  still lags  $\hat{\theta}_{\text{gen-EM}}$  by 10%, but we believe that more data can further reduce this gap.

Figure 4(d) shows that these trends are not particular to artificial data. On real WSJ data, the gap

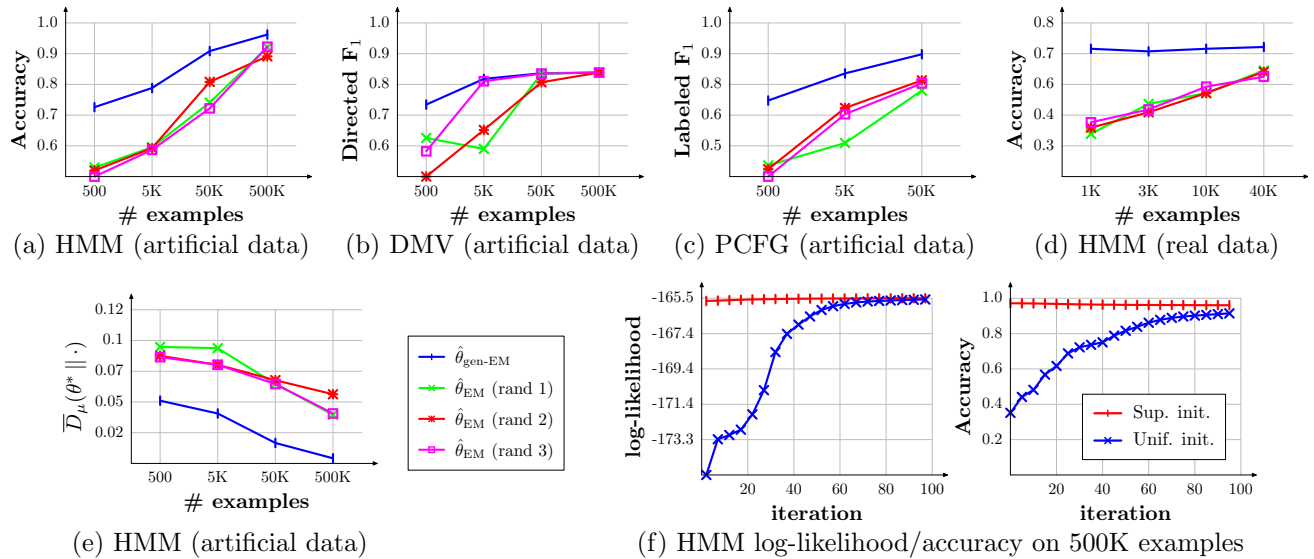


Figure 4: Compares the performance of  $\hat{\theta}_{EM}$  (EM with a uniform initialization) against  $\hat{\theta}_{gen-EM}$  (EM initialized with the supervised estimate) on (a–c) various models, (d) real data. (e) measures distance instead of accuracy and (f) shows a sample EM run.

between  $\hat{\theta}_{gen-EM}$  and  $\hat{\theta}_{EM}$  also diminishes for the HMM. To reaffirm the trends, we also measure distance  $D_\mu$ . Figure 4(e) shows that the distance from  $\hat{\theta}_{EM}$  to the true parameters  $\theta^*$  decreases, but the gap between  $\hat{\theta}_{gen-EM}$  and  $\hat{\theta}_{EM}$  does not close as decisively as it did for prediction error.

It is quite surprising that by simply running EM with a neutral initialization, we can accurately learn a complex model with thousands of parameters. Figures 4(f,g) show how both likelihood and accuracy, which both start quite low, improve substantially over time for the HMM on artificial data.

Carroll and Charniak (1992) report that EM fared poorly with local optima. We do not claim that there are no local optima, but only that the likelihood surface that EM is optimizing can become smoother with more examples. With more examples, there is less noise in the aggregate statistics, so it might be easier for EM to pick out the salient patterns.

Srebro et al. (2006) made a similar observation in the context of learning Gaussian mixtures. They characterized three regimes: one where EM was successful in recovering the true clusters (given lots of data), another where EM failed but the global optimum was successful, and the last where both failed (without much data).

There is also a rich body of theoretical work on

learning latent-variable models. Specialized algorithms can provably learn certain constrained discrete hidden-variable models, some in terms of weak generative capacity (Ron et al., 1998; Clark and Thollard, 2005; Adriaans, 1999), others in term of strong generative capacity (Dasgupta, 1999; Feldman et al., 2005). But with the exception of Dasgupta and Schulman (2007), there is little theoretical understanding of EM, let alone on complex model families such as the HMM, PCFG, and DMV.

## 8 Conclusion

In recent years, many methods have improved unsupervised induction, but these methods must still deal with the four types of errors we have identified in this paper. One of our main contributions of this paper is the idea of using the meta-model to diagnose the approximation error. Using this tool, we can better understand model biases and hopefully correct for them. We also introduced a method for measuring distances in face of label symmetry and ran experiments exploring the effectiveness of EM as a function of the amount of data. Finally, we hope that setting up the general framework to understand the errors of unsupervised induction systems will aid the development of better methods and further analyses.

## References

- P. W. Adriaans. 1999. Learning shallow context-free languages under simple distributions. Technical report, Stanford University.
- G. Carroll and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Workshop Notes for Statistically-Based NLP Techniques*, pages 1–13.
- A. Clark and F. Thollard. 2005. PAC-learnability of probabilistic deterministic finite state automata. *JMLR*, 5:473–497.
- A. Clark. 2001. Unsupervised induction of stochastic context free grammars with distributional clustering. In *CoNLL*.
- S. Dasgupta and L. Schulman. 2007. A probabilistic analysis of EM for mixtures of separated, spherical Gaussians. *JMLR*, 8.
- S. Dasgupta. 1999. Learning mixtures of Gaussians. In *FOCS*.
- J. Feldman, R. O’Donnell, and R. A. Servedio. 2005. Learning mixtures of product distributions over discrete domains. In *FOCS*, pages 501–510.
- S. Goldwater and T. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *ACL*.
- T. Grenager, D. Klein, and C. D. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *ACL*.
- A. Haghighi and D. Klein. 2006. Prototype-based grammar induction. In *ACL*.
- M. Johnson. 2007. Why doesn’t EM find good HMM POS-taggers? In *EMNLP/CoNLL*.
- D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.
- H. W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97.
- K. Kurihara and T. Sato. 2006. Variational Bayesian grammar induction for natural language. In *International Colloquium on Grammatical Inference*.
- B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20:155–171.
- F. Pereira and Y. Shabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL*.
- D. Ron, Y. Singer, and N. Tishby. 1998. On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and System Sciences*, 56:133–152.
- N. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL*.
- N. Smith and J. Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *ACL*.
- N. Srebro, G. Shakhnarovich, and S. Roweis. 2006. An investigation of computational and informational limits in Gaussian mixture clustering. In *ICML*, pages 865–872.

# Joint Word Segmentation and POS Tagging using a Single Perceptron

Yue Zhang and Stephen Clark

Oxford University Computing Laboratory

Wolfson Building, Parks Road

Oxford OX1 3QD, UK

{yue.zhang, stephen.clark}@comlab.ox.ac.uk

## Abstract

For Chinese POS tagging, word segmentation is a preliminary step. To avoid error propagation and improve segmentation by utilizing POS information, segmentation and tagging can be performed simultaneously. A challenge for this joint approach is the large combined search space, which makes efficient decoding very hard. Recent research has explored the integration of segmentation and POS tagging, by decoding under restricted versions of the full combined search space. In this paper, we propose a joint segmentation and POS tagging model that does not impose any hard constraints on the interaction between word and POS information. Fast decoding is achieved by using a novel multiple-beam search algorithm. The system uses a discriminative statistical model, trained using the generalized perceptron algorithm. The joint model gives an error reduction in segmentation accuracy of 14.6% and an error reduction in tagging accuracy of 12.2%, compared to the traditional pipeline approach.

## 1 Introduction

Since Chinese sentences do not contain explicitly marked word boundaries, word segmentation is a necessary step before POS tagging can be performed. Typically, a Chinese POS tagger takes segmented inputs, which are produced by a separate word segmentor. This two-step approach, however, has an obvious flaw of error propagation, since word segmentation errors cannot be corrected by the POS tagger. A better approach would be to utilize POS in-

formation to improve word segmentation. For example, the POS-word pattern “number word” + “个 (a common measure word)” can help in segmenting the character sequence “一个人” into the word sequence “一 (one) 个 (measure word) 人 (person)” instead of “一 (one) 个人 (personal; adj)”. Moreover, the comparatively rare POS pattern “number word” + “number word” can help to prevent segmenting a long number word into two words.

In order to avoid error propagation and make use of POS information for word segmentation, segmentation and POS tagging can be viewed as a single task: given a raw Chinese input sentence, the joint POS tagger considers all possible segmented and tagged sequences, and chooses the overall best output. A major challenge for such a joint system is the large search space faced by the decoder. For a sentence with  $n$  characters, the number of possible output sequences is  $O(2^{n-1} \cdot T^n)$ , where  $T$  is the size of the tag set. Due to the nature of the combined candidate items, decoding can be inefficient even with dynamic programming.

Recent research on Chinese POS tagging has started to investigate joint segmentation and tagging, reporting accuracy improvements over the pipeline approach. Various decoding approaches have been used to reduce the combined search space. Ng and Low (2004) mapped the joint segmentation and POS tagging task into a single character sequence tagging problem. Two types of tags are assigned to each character to represent its segmentation and POS. For example, the tag “b\_NN” indicates a character at the beginning of a noun. Using this method, POS features are allowed to interact with segmentation.

Since tagging is restricted to characters, the search space is reduced to  $O((4T)^n)$ , and beam search decoding is effective with a small beam size. However, the disadvantage of this model is the difficulty in incorporating whole word information into POS tagging. For example, the standard “word + POS tag” feature is not explicitly applicable. Shi and Wang (2007) introduced POS information to segmentation by reranking.  $N$ -best segmentation outputs are passed to a separately-trained POS tagger, and the best output is selected using the overall POS-segmentation probability score. In this system, the decoding for word segmentation and POS tagging are still performed separately, and exact inference for both is possible. However, the interaction between POS and segmentation is restricted by reranking: POS information is used to improve segmentation only for the  $N$  segmentor outputs.

In this paper, we propose a novel joint model for Chinese word segmentation and POS tagging, which does not limiting the interaction between segmentation and POS information in reducing the combined search space. Instead, a novel multiple beam search algorithm is used to do decoding efficiently. Candidate ranking is based on a discriminative joint model, with features being extracted from segmented words and POS tags simultaneously. The training is performed by a single generalized perceptron (Collins, 2002). In experiments with the Chinese Treebank data, the joint model gave an error reduction of 14.6% in segmentation accuracy and 12.2% in the overall segmentation and tagging accuracy, compared to the traditional pipeline approach. In addition, the overall results are comparable to the best systems in the literature, which exploit knowledge outside the training data, even though our system is fully data-driven.

Different methods have been proposed to reduce error propagation between pipelined tasks, both in general (Sutton et al., 2004; Daumé III and Marcu, 2005; Finkel et al., 2006) and for specific problems such as language modeling and utterance classification (Saraclar and Roark, 2005) and labeling and chunking (Shimizu and Haas, 2006). Though our model is built specifically for Chinese word segmentation and POS tagging, the idea of using the perceptron model to solve multiple tasks simultaneously can be generalized to other tasks.

1	word $w$
2	word bigram $w_1w_2$
3	single-character word $w$
4	a word of length $l$ with starting character $c$
5	a word of length $l$ with ending character $c$
6	space-separated characters $c_1$ and $c_2$
7	character bigram $c_1c_2$ in any word
8	the first / last characters $c_1 / c_2$ of any word
9	word $w$ immediately before character $c$
10	character $c$ immediately before word $w$
11	the starting characters $c_1$ and $c_2$ of two consecutive words
12	the ending characters $c_1$ and $c_2$ of two consecutive words
13	a word of length $l$ with previous word $w$
14	a word of length $l$ with next word $w$

Table 1: Feature templates for the baseline segmentor

## 2 The Baseline System

We built a two-stage baseline system, using the perceptron segmentation model from our previous work (Zhang and Clark, 2007) and the perceptron POS tagging model from Collins (2002). We use *baseline system* to refer to the system which performs segmentation first, followed by POS tagging (using the single-best segmentation); *baseline segmentor* to refer to the segmentor from (Zhang and Clark, 2007) which performs segmentation only; and *baseline POS tagger* to refer to the Collins tagger which performs POS tagging only (given segmentation). The features used by the baseline segmentor are shown in Table 1. The features used by the POS tagger, some of which are different to those from Collins (2002) and are specific to Chinese, are shown in Table 2.

The word segmentation features are extracted from word bigrams, capturing word, word length and character information in the context. The word length features are normalized, with those more than 15 being treated as 15.

The POS tagging features are based on contextual information from the tag trigram, as well as the neighboring three-word window. To reduce overfitting and increase the decoding speed, templates 4, 5, 6 and 7 only include words with less than 3 characters. Like the baseline segmentor, the baseline tagger also normalizes word length features.

1	tag $t$ with word $w$
2	tag bigram $t_1t_2$
3	tag trigram $t_1t_2t_3$
4	tag $t$ followed by word $w$
5	word $w$ followed by tag $t$
6	word $w$ with tag $t$ and previous character $c$
7	word $w$ with tag $t$ and next character $c$
8	tag $t$ on single-character word $w$ in character trigram $c_1wc_2$
9	tag $t$ on a word starting with char $c$
10	tag $t$ on a word ending with char $c$
11	tag $t$ on a word containing char $c$ (not the starting or ending character)
12	tag $t$ on a word starting with char $c_0$ and containing char $c$
13	tag $t$ on a word ending with char $c_0$ and containing char $c$
14	tag $t$ on a word containing repeated char $cc$
15	tag $t$ on a word starting with character category $g$
16	tag $t$ on a word ending with character category $g$

Table 2: Feature templates for the baseline POS tagger

Templates 15 and 16 in Table 2 are inspired by the CTBMorph feature templates in Tseng et al. (2005), which gave the most accuracy improvement in their experiments. Here the category of a character is the set of tags seen on the character during training. Other morphological features from Tseng et al. (2005) are not used because they require extra web corpora besides the training data.

During training, the baseline POS tagger stores special word-tag pairs into a *tag dictionary* (Ratnaparkhi, 1996). Such information is used by the decoder to prune unlikely tags. For each word occurring more than  $N$  times in the training data, the decoder can only assign a tag the word has been seen with in the training data. This method led to improvement in the decoding speed as well as the output accuracy for English POS tagging (Ratnaparkhi, 1996). Besides tags for frequent words, our baseline POS tagger also uses the tag dictionary to store closed-set tags (Xia, 2000) – those associated only with a limited number of Chinese words.

### 3 Joint Segmentation and Tagging Model

In this section, we build a joint word segmentation and POS tagging model that uses exactly the same source of information as the baseline system, by applying the feature templates from the baseline word segmentor and POS tagger. No extra knowledge is used by the joint model. However, because word segmentation and POS tagging are performed simultaneously, POS information participates in word segmentation.

#### 3.1 Formulation of the joint model

We formulate joint word segmentation and POS tagging as a single problem, which maps a raw Chinese sentence to a segmented and POS tagged output. Given an input sentence  $x$ , the output  $F(x)$  satisfies:

$$F(x) = \arg \max_{y \in \text{GEN}(x)} \text{Score}(y)$$

where  $\text{GEN}(x)$  represents the set of possible outputs for  $x$ .

$\text{Score}(y)$  is computed by a feature-based linear model. Denoting the global feature vector for the tagged sentence  $y$  with  $\Phi(y)$ , we have:

$$\text{Score}(y) = \Phi(y) \cdot \vec{w}$$

where  $\vec{w}$  is the parameter vector in the model. Each element in  $\vec{w}$  gives a weight to its corresponding element in  $\Phi(y)$ , which is the count of a particular feature over the whole sentence  $y$ . We calculate the  $\vec{w}$  value by supervised learning, using the averaged perceptron algorithm (Collins, 2002), given in Figure 1.<sup>1</sup>

We take the union of feature templates from the baseline segmentor (Table 1) and POS tagger (Table 2) as the feature templates for the joint system. All features are treated equally and processed together according to the linear model, regardless of whether they are from the baseline segmentor or tagger. In fact, most features from the baseline POS tagger, when used in the joint model, represent segmentation patterns as well. For example, the aforementioned pattern “number word” + “↑”, which is

<sup>1</sup>In order to provide a comparison for the perceptron algorithm we also tried  $\text{svm}^{\text{struct}}$  (Tsochantaridis et al., 2004) for parameter estimation, but this training method was prohibitively slow.

**Inputs:** training examples  $(x_i, y_i)$   
**Initialization:** set  $\vec{w} = 0$   
**Algorithm:**  
 for  $t = 1..T, i = 1..N$   
   calculate  $z_i = \arg \max_{y \in \text{GEN}(x_i)} \Phi(y) \cdot \vec{w}$   
   if  $z_i \neq y_i$   
      $\vec{w} = \vec{w} + \Phi(y_i) - \Phi(z_i)$   
**Outputs:**  $\vec{w}$

Figure 1: The perceptron learning algorithm

useful only for the POS “number word” in the baseline tagger, is also an effective indicator of the segmentation of the two words (especially “ $\uparrow$ ”) in the joint model.

### 3.2 The decoding algorithm

One of the main challenges for the joint segmentation and POS tagging system is the decoding algorithm. The speed and accuracy of the decoder is important for the perceptron learning algorithm, but the system faces a very large search space of combined candidates. Given the linear model and feature templates, exact inference is very hard even with dynamic programming.

Experiments with the standard beam-search decoder described in (Zhang and Clark, 2007) resulted in low accuracy. This beam search algorithm processes an input sentence incrementally. At each stage, the incoming character is combined with existing partial candidates in all possible ways to generate new partial candidates. An agenda is used to control the search space, keeping only the  $B$  best partial candidates ending with the current character. The algorithm is simple and efficient, with a linear time complexity of  $O(BTn)$ , where  $n$  is the size of input sentence, and  $T$  is the size of the tag set ( $T = 1$  for pure word segmentation). It worked well for word segmentation alone (Zhang and Clark, 2007), even with an agenda size as small as 8, and a simple beam search algorithm also works well for POS tagging (Ratnaparkhi, 1996). However, when applied to the joint model, it resulted in a reduction in segmentation accuracy (compared to the baseline segmentor) even with  $B$  as large as 1024.

One possible cause of the poor performance of the standard beam search method is the combined nature of the candidates in the search space. In the base-

**Input:** raw sentence  $sent$  – a list of characters  
**Variables:** candidate sentence  $item$  – a list of (word, tag) pairs;  
 maximum word-length record  $maxlen$  for each tag;  
 the agenda list  $agendas$ ;  
 the tag dictionary  $tagdict$ ;  
 $start\_index$  for current word;  
 $end\_index$  for current word  
**Initialization:**  $agendas[0] = [“”]$ ,  
 $agendas[i] = []$  ( $i! = 0$ )

**Algorithm:**  
 for  $end\_index = 1$  to  $sent.length$ :  
   foreach  $tag$ :  
     for  $start\_index =$   
        $\max(1, end\_index - maxlen[tag] + 1)$   
     to  $end\_index$ :  
        $word = sent[start\_index..end\_index]$   
       if  $(word, tag)$  consistent with  $tagdict$ :  
         for  $item \in agendas[start\_index - 1]$ :  
            $item_1 = item$   
            $item_1.append((word, tag))$   
            $agendas[end\_index].insert(item_1)$   
**Outputs:**  $agendas[sent.length].best\_item$

Figure 2: The decoding algorithm for the joint word segmentor and POS tagger

line POS tagger, candidates in the beam are tagged sequences ending with the current word, which can be compared directly with each other. However, for the joint problem, candidates in the beam are segmented and tagged sequences up to the current character, where the last word can be a complete word or a partial word. A problem arises in whether to give POS tags to incomplete words. If partial words are given POS tags, it is likely that some partial words are “justified” as complete words by the current POS information. On the other hand, if partial words are not given POS tag features, the correct segmentation for long words can be lost during partial candidate comparison (since many short completed words with POS tags are likely to be preferred to a long incomplete word with no POS tag features).<sup>2</sup>

<sup>2</sup>We experimented with both assigning POS features to partial words and omitting them; the latter method performed better but both performed significantly worse than the multiple beam search method described below.

Another possible cause is the exponential growth in the number of possible candidates with increasing sentence size. The number increases from  $O(T^n)$  for the baseline POS tagger to  $O(2^{n-1}T^n)$  for the joint system. As a result, for an incremental decoding algorithm, the number of possible candidates increases exponentially with the current word or character index. In the POS tagging problem, a new incoming word enlarges the number of possible candidates by a factor of  $T$  (the size of the tag set). For the joint problem, however, the enlarging factor becomes  $2T$  with each incoming character. The speed of search space expansion is much faster, but the number of candidates is still controlled by a single, fixed-size beam at any stage. If we assume that the beam is not large enough for all the candidates at each stage, then, from the newly generated candidates, the baseline POS tagger can keep  $1/T$  for the next processing stage, while the joint model can keep only  $1/2T$ , and has to discard the rest. Therefore, even when the candidate comparison standard is ignored, we can still see that the chance for the overall best candidate to fall out of the beam is largely increased. Since the search space growth is exponential, increasing the fixed beam size is not effective in solving the problem.

To solve the above problems, we developed a multiple beam search algorithm, which compares candidates only with complete tagged words, and enables the size of the search space to scale with the input size. The algorithm is shown in Figure 2. In this decoder, an agenda is assigned to each character in the input sentence, recording the  $B$  best segmented and tagged partial candidates ending with the character. The input sentence is still processed incrementally. However, now when a character is processed, existing partial candidates ending with any previous characters are available. Therefore, the decoder enumerates all possible tagged words ending with the current character, and combines each word with the partial candidates ending with its previous character. All input characters are processed in the same way, and the final output is the best candidate in the final agenda. The time complexity of the algorithm is  $O(WTBn)$ , with  $W$  being the maximum word size,  $T$  being the total number of POS tags and  $n$  the number of characters in the input. It is also linear in the input size. Moreover, the decoding algorithm

gives competent accuracy with a small agenda size of  $B = 16$ .

To further limit the search space, two optimizations are used. First, the maximum word length for each tag is recorded and used by the decoder to prune unlikely candidates. Because the majority of tags only apply to words with length 1 or 2, this method has a strong effect. Development tests showed that it improves the speed significantly, while having a very small negative influence on the accuracy. Second, like the baseline POS tagger, the tag dictionary is used for Chinese closed set tags and the tags for frequent words. To words outside the tag dictionary, the decoder still tries to assign every possible tag.

### 3.3 Online learning

Apart from features, the decoder maintains other types of information, including the tag dictionary, the word frequency counts used when building the tag dictionary, the maximum word lengths by tag, and the character categories. The above data can be collected by scanning the corpus before training starts. However, in both the baseline tagger and the joint POS tagger, they are updated incrementally during the perceptron training process, consistent with online learning.<sup>3</sup>

The online updating of word frequencies, maximum word lengths and character categories is straightforward. For the online updating of the tag dictionary, however, the decision for frequent words must be made dynamically because the word frequencies keep changing. This is done by caching the number of occurrences of the current most frequent word  $M$ , and taking all words currently above the threshold  $M/5000 + 5$  as frequent words. 5000 is a rough figure to control the number of frequent words, set according to Zipf’s law. The parameter 5 is used to force all tags to be enumerated before a word is seen more than 5 times.

## 4 Related Work

Ng and Low (2004) and Shi and Wang (2007) were described in the Introduction. Both models reduced

<sup>3</sup>We took this approach because we wanted the whole training process to be online. However, for comparison purposes, we also tried precomputing the above information before training and the difference in performance was negligible.



the large search space by imposing strong restrictions on the form of search candidates. In particular, Ng and Low (2004) used character-based POS tagging, which prevents some important POS tagging features such as word + POS tag; Shi and Wang (2007) used an  $N$ -best reranking approach, which limits the influence of POS tagging on segmentation to the  $N$ -best list. In comparison, our joint model does not impose any hard limitations on the interaction between segmentation and POS information.<sup>4</sup> Fast decoding speed is achieved by using a novel multiple-beam search algorithm.

Nakagawa and Uchimoto (2007) proposed a hybrid model for word segmentation and POS tagging using an HMM-based approach. Word information is used to process known-words, and character information is used for unknown words in a similar way to Ng and Low (2004). In comparison, our model handles character and word information simultaneously in a single perceptron model.

## 5 Experiments

The Chinese Treebank (CTB) 4 is used for the experiments. It is separated into two parts: CTB 3 (420K characters in 150K words / 10364 sentences) is used for the final 10-fold cross validation, and the rest (240K characters in 150K words / 4798 sentences) is used as training and test data for development.

The standard F-scores are used to measure both the word segmentation accuracy and the overall segmentation and tagging accuracy, where the overall accuracy is  $TF = 2pr / (p + r)$ , with the precision  $p$  being the percentage of correctly segmented and tagged words in the decoder output, and the recall  $r$  being the percentage of gold-standard tagged words that are correctly identified by the decoder. For direct comparison with Ng and Low (2004), the POS tagging accuracy is also calculated by the percentage of correct tags on each character.

### 5.1 Development experiments

The learning curves of the baseline and joint models are shown in Figure 3, Figure 4 and Figure 5, respectively. These curves are used to show the conver-

<sup>4</sup>Apart from the beam search algorithm, we do impose some minor limitations on the search space by methods such as the tag dictionary, but these can be seen as optional pruning methods for optimization.

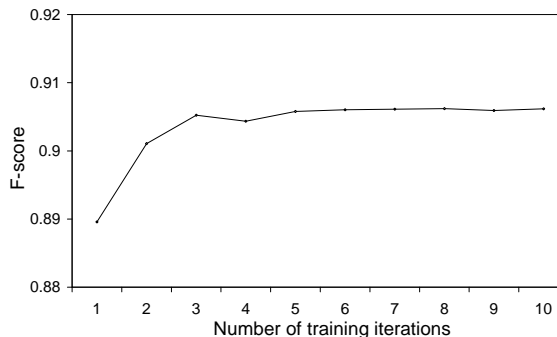


Figure 3: The learning curve of the baseline segmentor

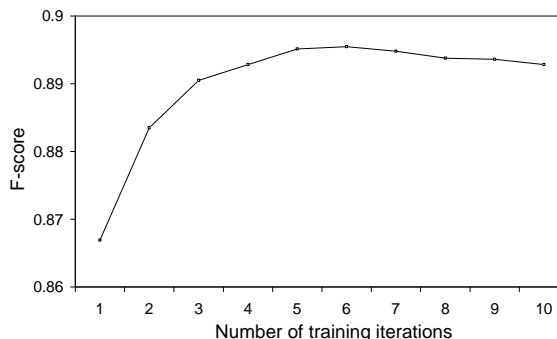


Figure 4: The learning curve of the baseline tagger

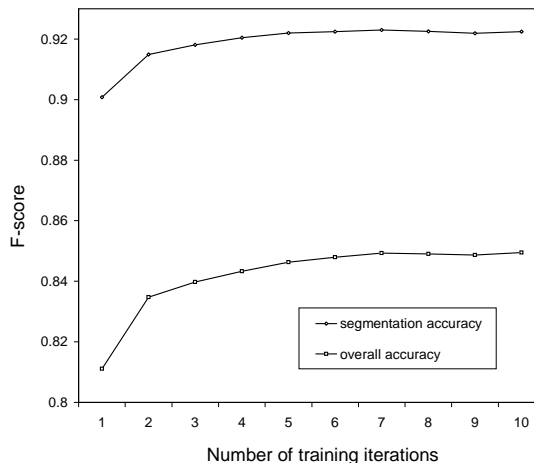


Figure 5: The learning curves of the joint system

gence of perceptron and decide the number of training iterations for the test. It should be noticed that the accuracies from Figure 4 and Figure 5 are not comparable because gold-standard segmentation is used as the input for the baseline tagger. According to the figures, the number of training iterations

Tag	Seg	NN	NR	VV	AD	JJ	CD
NN	20.47	–	0.78	4.80	0.67	2.49	0.04
NR	5.95	3.61	–	0.19	0.04	0.07	0
VV	12.13	6.51	0.11	–	0.93	0.56	0.04
AD	3.24	0.30	0	0.71	–	0.33	0.22
JJ	3.09	0.93	0.15	0.26	0.26	–	0.04
CD	1.08	0.04	0	0	0.07	0	–

Table 3: Error analysis for the joint model

for the baseline segmentor, POS tagger, and the joint system are set to 8, 6, and 7, respectively for the remaining experiments.

There are many factors which can influence the accuracy of the joint model. Here we consider the special character category features and the effect of the tag dictionary. The character category features (templates 15 and 16 in Table 2) represent a Chinese character by all the tags associated with the character in the training data. They have been shown to improve the accuracy of a Chinese POS tagger (Tseng et al., 2005). In the joint model, these features also represent segmentation information, since they concern the starting and ending characters of a word. Development tests showed that the overall tagging F-score of the joint model increased from 84.54% to 84.93% using the character category features. In the development test, the use of the tag dictionary improves the decoding speed of the joint model, reducing the decoding time from 416 seconds to 256 seconds. The overall tagging accuracy also increased slightly, consistent with observations from the pure POS tagger.

The error analysis for the development test is shown in Table 3. Here an error is counted when a word in the standard output is not produced by the decoder, due to incorrect segmentation or tag assignment. Statistics about the six most frequently mistaken tags are shown in the table, where each row presents the analysis of one tag from the standard output, and each column gives a wrongly assigned value. The column “Seg” represents segmentation errors. Each figure in the table shows the percentage of the corresponding error from all the errors.

It can be seen from the table that the NN-VV and VV-NN mistakes were the most commonly made by the decoder, while the NR-NN mistakes are also fre-

#	Baseline			Joint		
	<i>SF</i>	<i>TF</i>	<i>TA</i>	<i>SF</i>	<i>TF</i>	<i>TA</i>
1	96.98	92.91	94.14	97.21	93.46	94.66
2	97.16	93.20	94.34	97.62	93.85	94.79
3	95.02	89.53	91.28	95.94	90.86	92.38
4	95.51	90.84	92.55	95.92	91.60	93.31
5	95.49	90.91	92.57	96.06	91.72	93.25
6	93.50	87.33	89.87	94.56	88.83	91.14
7	94.48	89.44	91.61	95.30	90.51	92.41
8	93.58	88.41	90.93	95.12	90.30	92.32
9	93.92	89.15	91.35	94.79	90.33	92.45
10	96.31	91.58	93.01	96.45	91.96	93.45
Av.	95.20	90.33	92.17	95.90	91.34	93.02

Table 4: The accuracies by 10-fold cross validation

*SF* – segmentation F-score,  
*TF* – overall F-score,  
*TA* – tagging accuracy by character.

quent. These three types of errors significantly outnumber the rest, together contributing 14.92% of all the errors. Moreover, the most commonly mistaken tags are NN and VV, while among the most frequent tags in the corpus, PU, DEG and M had comparatively less errors. Lastly, segmentation errors contribute around half (51.47%) of all the errors.

## 5.2 Test results

10-fold cross validation is performed to test the accuracy of the joint word segmentor and POS tagger, and to make comparisons with existing models in the literature. Following Ng and Low (2004), we partition the sentences in CTB 3, ordered by sentence ID, into 10 groups evenly. In the  $n$ th test, the  $n$ th group is used as the testing data.

Table 4 shows the detailed results for the cross validation tests, each row representing one test. As can be seen from the table, the joint model outperforms the baseline system in each test.

Table 5 shows the overall accuracies of the baseline and joint systems, and compares them to the relevant models in the literature. The accuracy of each model is shown in a row, where “Ng” represents the models from Ng and Low (2004) and “Shi” represents the models from Shi and Wang (2007). Each accuracy measure is shown in a column, including the segmentation F-score (*SF*), the overall tagging

Model	<i>SF</i>	<i>TF</i>	<i>TA</i>
Baseline+ (Ng)	95.1	–	91.7
Joint+ (Ng)	95.2	–	91.9
Baseline+* (Shi)	95.85	91.67	–
Joint+* (Shi)	96.05	91.86	–
Baseline (ours)	95.20	90.33	92.17
Joint (ours)	95.90	91.34	93.02

Table 5: The comparison of overall accuracies by 10-fold cross validation using CTB

- + – knowledge about special characters,
- \* – knowledge from semantic net outside CTB.

F-score (*TF*) and the tagging accuracy by characters (*TA*). As can be seen from the table, our joint model achieved the largest improvement over the baseline, reducing the segmentation error by 14.58% and the overall tagging error by 12.18%.

The overall tagging accuracy of our joint model was comparable to but less than the joint model of Shi and Wang (2007). Despite the higher accuracy improvement from the baseline, the joint system did not give higher overall accuracy. One likely reason is that Shi and Wang (2007) included knowledge about special characters and semantic knowledge from web corpora (which may explain the higher baseline accuracy), while our system is completely data-driven. However, the comparison is indirect because our partitions of the CTB corpus are different. Shi and Wang (2007) also chunked the sentences before doing 10-fold cross validation, but used an uneven split. We chose to follow Ng and Low (2004) and split the sentences evenly to facilitate further comparison.

Compared with Ng and Low (2004), our baseline model gave slightly better accuracy, consistent with our previous observations about the word segmentors (Zhang and Clark, 2007). Due to the large accuracy gain from the baseline, our joint model performed much better.

In summary, when compared with existing joint word segmentation and POS tagging systems in the literature, our proposed model achieved the best accuracy boost from the cascaded baseline, and competent overall accuracy.

## 6 Conclusion and Future Work

We proposed a joint Chinese word segmentation and POS tagging model, which achieved a considerable reduction in error rate compared to a baseline two-stage system.

We used a single linear model for combined word segmentation and POS tagging, and chose the generalized perceptron algorithm for joint training. and beam search for efficient decoding. However, the application of beam search was far from trivial because of the size of the combined search space. Motivated by the question of what are the comparable partial hypotheses in the space, we developed a novel multiple beam search decoder which effectively explores the large search space. Similar techniques can potentially be applied to other problems involving joint inference in NLP.

Other choices are available for the decoding of a joint linear model, such as exact inference with dynamic programming, provided that the range of features allows efficient processing. The baseline feature templates for Chinese segmentation and POS tagging, when added together, makes exact inference for the proposed joint model very hard. However, the accuracy loss from the beam decoder, as well as alternative decoding algorithms, are worth further exploration.

The joint system takes features only from the baseline segmentor and the baseline POS tagger to allow a fair comparison. There may be additional features that are particularly useful to the joint system. Open features, such as knowledge of numbers and European letters, and relationships from semantic networks (Shi and Wang, 2007), have been reported to improve the accuracy of segmentation and POS tagging. Therefore, given the flexibility of the feature-based linear model, an obvious next step is the study of open features in the joint segmentor and POS tagger.

## Acknowledgements

We thank Hwee-Tou Ng and Mengqiu Wang for their helpful discussions and sharing of experimental data, and the anonymous reviewers for their suggestions. This work is supported by the ORS and Clarendon Fund.

## References

- Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP conference*, pages 1–8, Philadelphia, PA.
- Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the ICML Conference*, pages 169–176, Bonn, Germany.
- Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the EMNLP Conference*, pages 618–626, Sydney, Australia.
- Tetsuji Nakagawa and Kiyotaka Uchimoto. 2007. A hybrid approach to word segmentation and pos tagging. In *Proceedings of ACL Demo and Poster Session*, pages 217–220, Prague, Czech Republic.
- Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? Word-based or character-based? In *Proceedings of the EMNLP Conference*, pages 277–284, Barcelona, Spain.
- Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.
- Murat Saraclar and Brian Roark. 2005. Joint discriminative language modeling and utterance classification. In *Proceedings of the ICASSP Conference*, volume 1, Philadelphia, USA.
- Yanxin Shi and Mengqiu Wang. 2007. A dual-layer CRF based joint decoding method for cascade segmentation and labelling tasks. In *Proceedings of the IJCAI Conference*, Hyderabad, India.
- Nobuyuki Shimizu and Andrew Haas. 2006. Exact decoding for jointly labeling and chunking sequences. In *Proceedings of the COLING/ACL Conference, Poster Sessions*, Sydney, Australia.
- Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of the ICML Conference*, Banff, Canada.
- Huihsin Tseng, Daniel Jurafsky, and Christopher Manning. 2005. Morphological features help POS tagging of unknown words across language varieties. In *Proceedings of the Fourth SIGHAN Workshop*, Jeju Island, Korea.
- I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the ICML Conference*, Banff, Canada.
- Fei Xia. 2000. The part-of-speech tagging guidelines for the Chinese Treebank (3.0). *IRCS Report*, University of Pennsylvania.
- Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the ACL Conference*, pages 840–847, Prague, Czech Republic.

# A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging

Wenbin Jiang<sup>†</sup>

Liang Huang<sup>‡</sup>

Qun Liu<sup>†</sup>

Yajuan Lü<sup>†</sup>

<sup>†</sup>Key Lab. of Intelligent Information Processing  
Institute of Computing Technology  
Chinese Academy of Sciences  
P.O. Box 2704, Beijing 100190, China  
jiangwenbin@ict.ac.cn

<sup>‡</sup>Department of Computer & Information Science  
University of Pennsylvania  
Levine Hall, 3330 Walnut Street  
Philadelphia, PA 19104, USA  
lhuang3@cis.upenn.edu

## Abstract

We propose a cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. With a character-based perceptron as the core, combined with real-valued features such as language models, the cascaded model is able to efficiently utilize knowledge sources that are inconvenient to incorporate into the perceptron directly. Experiments show that the cascaded model achieves improved accuracies on both segmentation only and joint segmentation and part-of-speech tagging. On the Penn Chinese Treebank 5.0, we obtain an error reduction of 18.5% on segmentation and 12% on joint segmentation and part-of-speech tagging over the perceptron-only baseline.

## 1 Introduction

Word segmentation and part-of-speech (POS) tagging are important tasks in computer processing of Chinese and other Asian languages. Several models were introduced for these problems, for example, the Hidden Markov Model (HMM) (Rabiner, 1989), Maximum Entropy Model (ME) (Ratnaparkhi and Adwait, 1996), and Conditional Random Fields (CRFs) (Lafferty et al., 2001). CRFs have the advantage of flexibility in representing features compared to generative ones such as HMM, and usually behaves the best in the two tasks. Another widely used discriminative method is the perceptron algorithm (Collins, 2002), which achieves comparable performance to CRFs with much faster training, so we base this work on the perceptron.

To segment and tag a character sequence, there are two strategies to choose: performing POS tagging following segmentation; or joint segmentation and POS tagging (Joint S&T). Since the typical approach of discriminative models treats segmentation as a labelling problem by assigning each character a boundary tag (Xue and Shen, 2003), Joint S&T can be conducted in a labelling fashion by expanding boundary tags to include POS information (Ng and Low, 2004). Compared to performing segmentation and POS tagging one at a time, Joint S&T can achieve higher accuracy not only on segmentation but also on POS tagging (Ng and Low, 2004). Besides the usual character-based features, additional features dependent on POS's or words can also be employed to improve the performance. However, as such features are generated dynamically during the decoding procedure, two limitations arise: on the one hand, the amount of parameters increases rapidly, which is apt to overfit on training corpus; on the other hand, exact inference by dynamic programming is intractable because the current predication relies on the results of prior predications. As a result, many theoretically useful features such as higher-order word or POS  $n$ -grams are difficult to be incorporated in the model efficiently.

To cope with this problem, we propose a cascaded linear model inspired by the log-linear model (Och and Ney, 2004) widely used in statistical machine translation to incorporate different kinds of knowledge sources. Shown in Figure 1, the cascaded model has a two-layer architecture, with a character-based perceptron as the core combined with other real-valued features such as language models. We

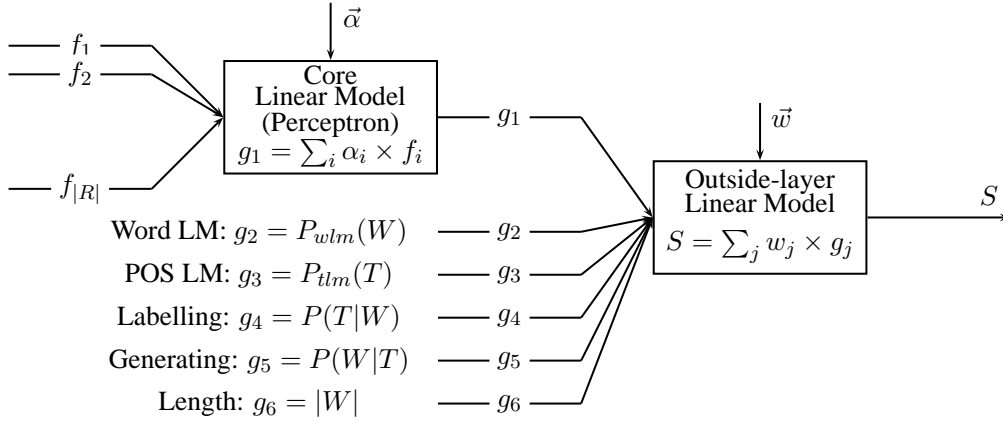


Figure 1: Structure of Cascaded Linear Model.  $|R|$  denotes the scale of the feature space of the core perceptron.

will describe it in detail in Section 4. In this architecture, knowledge sources that are intractable to incorporate into the perceptron, can be easily incorporated into the outside linear model. In addition, as these knowledge sources are regarded as separate features, we can train their corresponding models independently with each other. This is an interesting approach when the training corpus is large as it reduces the time and space consumption. Experiments show that our cascaded model can utilize different knowledge sources effectively and obtain accuracy improvements on both segmentation and Joint S&T.

## 2 Segmentation and POS Tagging

Given a Chinese character sequence:

$$C_{1:n} = C_1 C_2 \dots C_n$$

the segmentation result can be depicted as:

$$C_{1:e_1} C_{e_1+1:e_2} \dots C_{e_{m-1}+1:e_m}$$

while the segmentation and POS tagging result can be depicted as:

$$C_{1:e_1}/t_1 C_{e_1+1:e_2}/t_2 \dots C_{e_{m-1}+1:e_m}/t_m$$

Here,  $C_i$  ( $i = 1..n$ ) denotes Chinese character,  $t_i$  ( $i = 1..m$ ) denotes POS tag, and  $C_{l:r}$  ( $l \leq r$ ) denotes character sequence ranges from  $C_l$  to  $C_r$ . We can see that segmentation and POS tagging task is to divide a character sequence into several subsequences and label each of them a POS tag.

It is a better idea to perform segmentation and POS tagging jointly in a uniform framework. According to Ng and Low (2004), the segmentation

task can be transformed to a tagging problem by assigning each character a boundary tag of the following four types:

- $b$ : the begin of the word
- $m$ : the middle of the word
- $e$ : the end of the word
- $s$ : a single-character word

We can extract segmentation result by splitting the labelled result into subsequences of pattern  $s$  or  $bm^*e$  which denote single-character word and multi-character word respectively. In order to perform POS tagging at the same time, we expand boundary tags to include POS information by attaching a POS to the tail of a boundary tag as a postfix following Ng and Low (2004). As each tag is now composed of a boundary part and a POS part, the joint S&T problem is transformed to a uniform boundary-POS labelling problem. A subsequence of boundary-POS labelling result indicates a word with POS  $t$  only if the boundary tag sequence composed of its boundary part conforms to  $s$  or  $bm^*e$  style, and all POS tags in its POS part equal to  $t$ . For example, a tag sequence  $b\_NN m\_NN e\_NN$  represents a three-character word with POS tag  $NN$ .

## 3 The Perceptron

The perceptron algorithm introduced into NLP by Collins (2002), is a simple but effective discriminative training method. It has comparable performance

Non-lexical-target	Instances
$C_n (n = -2..2)$	$C_{-2}=\text{下}, C_{-1}=\text{雨}, C_0=\text{天}, C_1=\text{地}, C_2=\text{面}$
$C_n C_{n+1} (n = -2..1)$	$C_{-2}C_{-1}=\text{下雨}, C_{-1}C_0=\text{雨天}, C_0C_1=\text{天地}, C_1C_2=\text{地面}$
$C_{-1}C_1$	$C_{-1}C_1=\text{雨地}$
Lexical-target	Instances
$C_0 C_n (n = -2..2)$	$C_0 C_{-2}=\text{天下}, C_0 C_{-1}=\text{天雨}, C_0 C_0=\text{天天}, C_0 C_1=\text{天地}, C_0 C_2=\text{天面}$
$C_0 C_n C_{n+1} (n = -2..1)$	$C_0 C_{-2} C_{-1}=\text{天天下雨}, C_0 C_{-1} C_0=\text{天雨天}, C_0 C_0 C_1=\text{天天地}, C_0 C_1 C_2=\text{天地面}$
$C_0 C_{-1} C_1$	$C_0 C_{-1} C_1=\text{天雨地}$

Table 1: Feature templates and instances. Suppose we are considering the third character "天" in "下雨 天 地面".

to CRFs, while with much faster training. The perceptron has been used in many NLP tasks, such as POS tagging (Collins, 2002), Chinese word segmentation (Ng and Low, 2004; Zhang and Clark, 2007) and so on. We trained a character-based perceptron for Chinese Joint S&T, and found that the perceptron itself could achieve considerably high accuracy on segmentation and Joint S&T. In following subsections, we describe the feature templates and the perceptron training algorithm.

### 3.1 Feature Templates

The feature templates we adopted are selected from those of Ng and Low (2004). To compare with others conveniently, we excluded the ones forbidden by the close test regulation of SIGHAN, for example,  $Pu(C_0)$ , indicating whether character  $C_0$  is a punctuation.

All feature templates and their instances are shown in Table 1.  $C$  represents a Chinese character while the subscript of  $C$  indicates its position in the sentence relative to the current character (it has the subscript 0). Templates immediately borrowed from Ng and Low (2004) are listed in the upper column named *non-lexical-target*. We called them *non-lexical-target* because predications derived from them can predicate without considering the current character  $C_0$ . Templates in the column below are expanded from the upper ones. We add a field  $C_0$  to each template in the upper column, so that it can carry out predication according to not only the context but also the current character itself. As predications generated from such templates depend on the current character, we name these templates *lexical-target*. Note that the templates of Ng and Low (2004) have already contained some *lexical-target* ones. With the two kinds

---

### Algorithm 1 Perceptron training algorithm.

---

- 1: **Input:** Training examples  $(x_i, y_i)$
  - 2:  $\vec{\alpha} \leftarrow \mathbf{0}$
  - 3: **for**  $t \leftarrow 1 \dots T$  **do**
  - 4:     **for**  $i \leftarrow 1 \dots N$  **do**
  - 5:          $z_i \leftarrow \operatorname{argmax}_{z \in \text{GEN}(x_i)} \Phi(x_i, z) \cdot \vec{\alpha}$
  - 6:         **if**  $z_i \neq y_i$  **then**
  - 7:              $\vec{\alpha} \leftarrow \vec{\alpha} + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
  - 8: **Output:** Parameters  $\vec{\alpha}$
- 

of predications, the perceptron model will do exact predicating to the best of its ability, and can back off to approximately predicating if exact predicating fails.

### 3.2 Training Algorithm

We adopt the perceptron training algorithm of Collins (2002) to learn a discriminative model mapping from inputs  $x \in X$  to outputs  $y \in Y$ , where  $X$  is the set of sentences in the training corpus and  $Y$  is the set of corresponding labelled results. Following Collins, we use a function  $\text{GEN}(x)$  generating all candidate results of an input  $x$ , a representation  $\Phi$  mapping each training example  $(x, y) \in X \times Y$  to a feature vector  $\Phi(x, y) \in R^d$ , and a parameter vector  $\vec{\alpha} \in R^d$  corresponding to the feature vector.  $d$  means the dimension of the vector space, it equals to the amount of features in the model. For an input character sequence  $x$ , we aim to find an output  $F(x)$  satisfying:

$$F(x) = \operatorname{argmax}_{y \in \text{GEN}(x)} \Phi(x, y) \cdot \vec{\alpha} \quad (1)$$

$\Phi(x, y) \cdot \vec{\alpha}$  represents the inner product of feature vector  $\Phi(x, y)$  and the parameter vector  $\vec{\alpha}$ . We used the algorithm depicted in Algorithm 1 to tune the parameter vector  $\vec{\alpha}$ .

To alleviate overfitting on the training examples, we use the refinement strategy called “averaged parameters” (Collins, 2002) to the algorithm in Algorithm 1.

## 4 Cascaded Linear Model

In theory, any useful knowledge can be incorporated into the perceptron directly, besides the character-based features already adopted. Additional features most widely used are related to word or POS  $n$ -grams. However, such features are generated dynamically during the decoding procedure so that the feature space enlarges much more rapidly. Figure 2 shows the growing tendency of feature space with the introduction of these features as well as the character-based ones. We noticed that the templates related to word unigrams and bigrams bring to the feature space an enlargement much rapider than the character-base ones, not to mention the higher-order grams such as trigrams or 4-grams. In addition, even though these higher grams were managed to be used, there still remains another problem: as the current predication relies on the results of prior ones, the decoding procedure has to resort to approximate inference by maintaining a list of  $N$ -best candidates at each predication position, which evokes a potential risk to depress the training.

To alleviate the drawbacks, we propose a cascaded linear model. It has a two-layer architecture, with a perceptron as the core and another linear model as the outside-layer. Instead of incorporating all features into the perceptron directly, we first trained the perceptron using character-based features, and several other sub-models using additional ones such as word or POS  $n$ -grams, then trained the outside-layer linear model using the outputs of these sub-models, including the perceptron. Since the perceptron is fixed during the second training step, the whole training procedure need relative small time and memory cost.

The outside-layer linear model, similar to those in SMT, can synthetically utilize different knowledge sources to conduct more accurate comparison between candidates. In this layer, each knowledge source is treated as a feature with a corresponding weight denoting its relative importance. Suppose we have  $n$  features  $g_j$  ( $j = 1..n$ ) coupled with  $n$  corre-

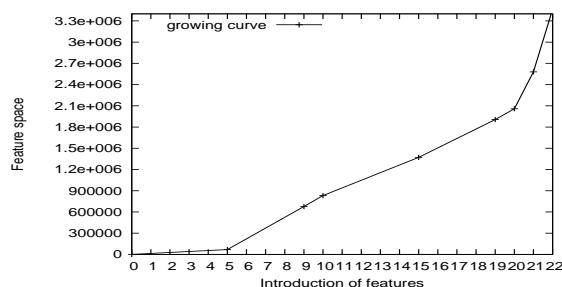


Figure 2: Feature space growing curve. The horizontal scope  $X[i:j]$  denotes the introduction of different templates.  $X[0:5]$ :  $C_n$  ( $n = -2..2$ );  $X[5:9]$ :  $C_n C_{n+1}$  ( $n = -2..1$ );  $X[9:10]$ :  $C_{-1} C_1$ ;  $X[10:15]$ :  $C_0 C_n$  ( $n = -2..2$ );  $X[15:19]$ :  $C_0 C_n C_{n+1}$  ( $n = -2..1$ );  $X[19:20]$ :  $C_0 C_{-1} C_1$ ;  $X[20:21]$ :  $W_0$ ;  $X[21:22]$ :  $W_{-1} W_0$ .  $W_0$  denotes the current considering word, while  $W_{-1}$  denotes the word in front of  $W_0$ . All the data are collected from the training procedure on MSR corpus of SIGHAN bake-off 2.

sponding weights  $w_j$  ( $j = 1..n$ ), each feature  $g_j$  gives a score  $g_j(r)$  to a candidate  $r$ , then the total score of  $r$  is given by:

$$S(r) = \sum_{j=1..n} w_j \times g_j(r) \quad (2)$$

The decoding procedure aims to find the candidate  $r^*$  with the highest score:

$$r^* = \underset{r}{\operatorname{argmax}} S(r) \quad (3)$$

While the mission of the training procedure is to tune the weights  $w_j$  ( $j = 1..n$ ) to guarantee that the candidate  $r$  with the highest score happens to be the best result with a high probability.

As all the sub-models, including the perceptron, are regarded as separate features of the outside-layer linear model, we can train them respectively with special algorithms. In our experiments we trained a 3-gram word language model measuring the fluency of the segmentation result, a 4-gram POS language model functioning as the product of state-transition probabilities in HMM, and a word-POS co-occurrence model describing how much probably a word sequence coexists with a POS sequence. As shown in Figure 1, the character-based perceptron is used as the inside-layer linear model and sends its output to the outside-layer. Besides the output of the perceptron, the outside-layer also receive the outputs



of the word LM, the POS LM, the co-occurrence model and a word count penalty which is similar to the translation length penalty in SMT.

#### 4.1 Language Model

Language model (LM) provides linguistic probabilities of a word sequence. It is an important measure of fluency of the translation in SMT. Formally, an  $n$ -gram word LM approximates the probability of a word sequence  $W = w_{1:m}$  with the following product:

$$P_{wlm}(W) = \prod_{i=1}^m \Pr(w_i | w_{\max(0, i-n+1):i-1}) \quad (4)$$

Similarly, the  $n$ -gram POS LM of a POS sequence  $T = t_{1:m}$  is:

$$P_{tlm}(T) = \prod_{i=1}^m \Pr(t_i | t_{\max(0, i-n+1):i-1}) \quad (5)$$

Notice that a bi-gram POS LM functions as the product of transition probabilities in HMM.

#### 4.2 Word-POS Co-occurrence Model

Given a training corpus with POS tags, we can train a word-POS co-occurrence model to approximate the probability that the word sequence of the labelled result co-exists with its corresponding POS sequence. Using  $W = w_{1:m}$  to denote the word sequence,  $T = t_{1:m}$  to denote the corresponding POS sequence,  $P(T|W)$  to denote the probability that  $W$  is labelled as  $T$ , and  $P(W|T)$  to denote the probability that  $T$  generates  $W$ , we can define the co-occurrence model as follows:

$$Co(W, T) = P(T|W)^{\lambda_{wt}} \times P(W|T)^{\lambda_{tw}} \quad (6)$$

$\lambda_{wt}$  and  $\lambda_{tw}$  denote the corresponding weights of the two components.

Suppose the conditional probability  $Pr(t|w)$  describes the probability that the word  $w$  is labelled as the POS  $t$ , while  $Pr(w|t)$  describes the probability that the POS  $t$  generates the word  $w$ , then  $P(T|W)$  can be approximated by:

$$P(T|W) \simeq \prod_{k=1}^m \Pr(t_k | w_k) \quad (7)$$

And  $P(W|T)$  can be approximated by:

$$P(W|T) \simeq \prod_{k=1}^m \Pr(w_k | t_k) \quad (8)$$

$Pr(w|t)$  and  $Pr(t|w)$  can be easily acquired by Maximum Likelihood Estimates (MLE) over the corpus. For instance, if the word  $w$  appears  $N$  times in training corpus and is labelled as POS  $t$  for  $n$  times, the probability  $Pr(t|w)$  can be estimated by the formula below:

$$Pr(t|w) \simeq \frac{n}{N} \quad (9)$$

The probability  $Pr(w|t)$  could be estimated through the same approach.

To facilitate tuning the weights, we use two components of the co-occurrence model  $Co(W, T)$  to represent the co-occurrence probability of  $W$  and  $T$ , rather than use  $Co(W, T)$  itself. In the rest of the paper, we will call them labelling model and generating model respectively.

### 5 Decoder

Sequence segmentation and labelling problem can be solved through a viterbi style decoding procedure. In Chinese Joint S&T, the mission of the decoder is to find the boundary-POS labelled sequence with the highest score. Given a Chinese character sequence  $C_{1:n}$ , the decoding procedure can proceed in a left-right fashion with a dynamic programming approach. By maintaining a stack of size  $N$  at each position  $i$  of the sequence, we can preserve the top  $N$  best candidate labelled results of subsequence  $C_{1:i}$  during decoding. At each position  $i$ , we enumerate all possible word-POS pairs by assigning each POS to each possible word formed from the character subsequence spanning length  $l = 1.. \min(i, K)$  ( $K$  is assigned 20 in all our experiments) and ending at position  $i$ , then we derive all candidate results by attaching each word-POS pair  $p$  (of length  $l$ ) to the tail of each candidate result at the prior position of  $p$  (position  $i-l$ ), and select for position  $i$  a  $N$ -best list of candidate results from all these candidates. When we derive a candidate result from a word-POS pair  $p$  and a candidate  $q$  at prior position of  $p$ , we calculate the scores of the word LM, the POS LM, the labelling probability and the generating probability,

---

**Algorithm 2** Decoding algorithm.

---

```
1: Input: character sequence  $C_{1:n}$ 
2: for  $i \leftarrow 1 .. n$  do
3:    $\mathcal{L} \leftarrow \emptyset$ 
4:   for  $l \leftarrow 1 .. \min(i, K)$  do
5:      $w \leftarrow C_{i-l+1:i}$ 
6:     for  $t \in POS$  do
7:        $p \leftarrow \text{label } w \text{ as } t$ 
8:       for  $q \in \mathcal{V}[i-l]$  do
9:         append  $D(q, p)$  to  $\mathcal{L}$ 
10:  sort  $\mathcal{L}$ 
11:   $\mathcal{V}[i] \leftarrow \mathcal{L}[1 : N]$ 
12: Output: n-best results  $\mathcal{V}[n]$ 
```

---

as well as the score of the perceptron model. In addition, we add the score of the word count penalty as another feature to alleviate the tendency of LMs to favor shorter candidates. By equation 2, we can synthetically evaluate all these scores to perform more accurately comparing between candidates.

Algorithm 2 shows the decoding algorithm. Lines 3 – 11 generate a  $N$ -best list for each character position  $i$ . Line 4 scans words of all possible lengths  $l$  ( $l = 1.. \min(i, K)$ , where  $i$  points to the current considering character). Line 6 enumerates all POS's for the word  $w$  spanning length  $l$  and ending at position  $i$ . Line 8 considers each candidate result in  $N$ -best list at prior position of the current word. Function  $D$  derives the candidate result from the word-POS pair  $p$  and the candidate  $q$  at prior position of  $p$ .

## 6 Experiments

We reported results from two set of experiments. The first was conducted to test the performance of the perceptron on segmentation on the corpus from SIGHAN Bakeoff 2, including the Academia Sinica Corpus (AS), the Hong Kong City University Corpus (CityU), the Peking University Corpus (PKU) and the Microsoft Research Corpus (MSR). The second was conducted on the Penn Chinese Treebank 5.0 (CTB5.0) to test the performance of the cascaded model on segmentation and Joint S&T. In all experiments, we use the averaged parameters for the perceptrons, and F-measure as the accuracy measure. With precision  $P$  and recall  $R$ , the balance F-measure is defined as:  $F = 2PR/(P + R)$ .

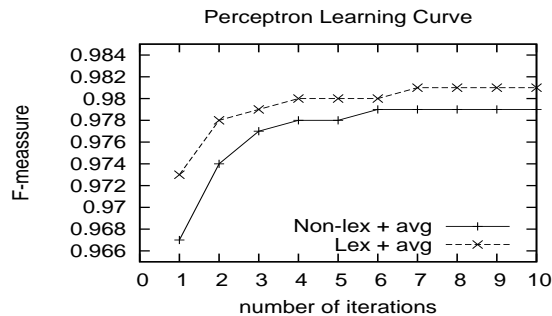


Figure 3: Averaged perceptron learning curves with Non-lexical-target and Lexical-target feature templates.

	AS	CityU	PKU	MSR
SIGHAN best	0.952	0.943	0.950	0.964
Zhang & Clark	0.946	0.951	0.945	0.972
our model	0.954	0.958	0.940	0.975

Table 2: F-measure on SIGHAN bakeoff 2. SIGHAN best: best scores SIGHAN reported on the four corpus, cited from Zhang and Clark (2007).

### 6.1 Experiments on SIGHAN Bakeoff

For convenience of comparing with others, we focus only on the close test, which means that any extra resource is forbidden except the designated training corpus. In order to test the performance of the *lexical-target* templates and meanwhile determine the best iterations over the training corpus, we randomly chosen 2,000 shorter sentences (less than 50 words) as the development set and the rest as the training set (84,294 sentences), then trained a perceptron model named NON-LEX using only *non-lexical-target* features and another named LEX using both the two kinds of features. Figure 3 shows their learning curves depicting the F-measure on the development set after 1 to 10 training iterations. We found that LEX outperforms NON-LEX with a margin of about 0.002 at each iteration, and its learning curve reaches a tableland at iteration 7. Then we trained LEX on each of the four corpora for 7 iterations. Test results listed in Table 2 shows that this model obtains higher accuracy than the best of SIGHAN Bakeoff 2 in three corpora (AS, CityU and MSR). On the three corpora, it also outperformed the word-based perceptron model of Zhang and Clark (2007). However, the accuracy on PKU corpus is obvious lower than the best score SIGHAN

Training setting	Test task	F-measure
POS-	Segmentation	0.971
POS+	Segmentation	0.973
POS+	Joint S&T	0.925

Table 3: F-measure on segmentation and Joint S&T of perceptrons. POS-: perceptron trained without POS, POS+: perceptron trained with POS.

reported, we need to conduct further research on this problem.

## 6.2 Experiments on CTB5.0

We turned to experiments on CTB 5.0 to test the performance of the cascaded model. According to the usual practice in syntactic analysis, we choose chapters 1 – 260 (18074 sentences) as training set, chapter 271 – 300 (348 sentences) as test set and chapter 301 – 325 (350 sentences) as development set.

At the first step, we conducted a group of contrasting experiments on the core perceptron, the first concentrated on the segmentation regardless of the POS information and reported the F-measure on segmentation only, while the second performed Joint S&T using POS information and reported the F-measure both on segmentation and on Joint S&T. Note that the accuracy of Joint S&T means that a word-POS pair is recognized only if both the boundary tags and the POS’s are correctly labelled.

The evaluation results are shown in Table 3. We find that Joint S&T can also improve the segmentation accuracy. However, the F-measure on Joint S&T is obvious lower, about a rate of 95% to the F-measure on segmentation. Similar trend appeared in experiments of Ng and Low (2004), where they conducted experiments on CTB 3.0 and achieved F-measure 0.919 on Joint S&T, a ratio of 96% to the F-measure 0.952 on segmentation.

As the next step, a group of experiments were conducted to investigate how well the cascaded linear model performs. Here the core perceptron was just the POS+ model in experiments above. Besides this perceptron, other sub-models are trained and used as additional features of the outside-layer linear model. We used SRI Language Modelling Toolkit (Stolcke and Andreas, 2002) to train a 3-gram word LM with modified Kneser-Ney smoothing (Chen and Goodman, 1998), and a 4-gram POS

Features	Segmentation F1	Joint S&T F1
All	0.9785	0.9341
All - PER	0.9049	0.8432
All - WLM	0.9785	0.9340
All - PLM	0.9752	0.9270
All - GPR	0.9774	0.9329
All - LPR	0.9765	0.9321
All - LEN	0.9772	0.9325

Table 4: Contribution of each feature. ALL: all features, PER: perceptron model, WLM: word language model, PLM: POS language model, GPR: generating model, LPR: labelling model, LEN: word count penalty.

LM with Witten-Bell smoothing, and we trained a word-POS co-occurrence model simply by MLE without smoothing. To obtain their corresponding weights, we adapted the minimum-error-rate training algorithm (Och, 2003) to train the outside-layer model. In order to inspect how much improvement each feature brings into the cascaded model, every time we removed a feature while retaining others, then retrained the model and tested its performance on the test set.

Table 4 shows experiments results. We find that the cascaded model achieves a F-measure increment of about 0.5 points on segmentation and about 0.9 points on Joint S&T, over the perceptron-only model POS+. We also find that the perceptron model functions as the kernel of the outside-layer linear model. Without the perceptron, the cascaded model (if we can still call it “cascaded”) performs poorly on both segmentation and Joint S&T. Among other features, the 4-gram POS LM plays the most important role, removing this feature causes F-measure decrement of 0.33 points on segmentation and 0.71 points on Joint S&T. Another important feature is the labelling model. Without it, the F-measure on segmentation and Joint S&T both suffer a decrement of 0.2 points. The generating model, which functions as that in HMM, brings an improvement of about 0.1 points to each test item. However unlike the three features, the word LM brings very tiny improvement. We suppose that the character-based features used in the perceptron play a similar role as the lower-order word LM, and it would be helpful if we train a higher-order word LM on a larger scale corpus. Finally, the word count penalty gives improvement to the cascaded model, 0.13 points on segmentation

and 0.16 points on Joint S&T.

In summary, the cascaded model can utilize these knowledge sources effectively, without causing the feature space of the perceptron becoming even larger. Experimental results show that, it achieves obvious improvement over the perceptron-only model, about from 0.973 to 0.978 on segmentation, and from 0.925 to 0.934 on Joint S&T, with error reductions of 18.5% and 12% respectively.

## 7 Conclusions

We proposed a cascaded linear model for Chinese Joint S&T. Under this model, many knowledge sources that may be intractable to be incorporated into the perceptron directly, can be utilized effectively in the outside-layer linear model. This is a substitute method to use both local and non-local features, and it would be especially useful when the training corpus is very large.

However, can the perceptron incorporate all the knowledge used in the outside-layer linear model? If this cascaded linear model were chosen, could more accurate generative models (LMs, word-POS co-occurrence model) be obtained by training on large scale corpus even if the corpus is not correctly labelled entirely, or by self-training on raw corpus in a similar approach to that of McClosky (2006)? In addition, all knowledge sources we used in the core perceptron and the outside-layer linear model come from the training corpus, whereas many open knowledge sources (lexicon etc.) can be used to improve performance (Ng and Low, 2004). How can we utilize these knowledge sources effectively? We will investigate these problems in the following work.

## Acknowledgement

This work was done while L. H. was visiting CAS/ICT. The authors were supported by National Natural Science Foundation of China, Contracts 60736014 and 60573188, and 863 State Key Project No. 2006AA010108 (W. J., Q. L., and Y. L.), and by NSF ITR EIA-0205456 (L. H.). We would also like to Hwee-Tou Ng for sharing his code, and Yang Liu and Yun Huang for suggestions.

## References

- Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.
- Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8, Philadelphia, USA.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th ICML*, pages 282–289, Massachusetts, USA.
- David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of ACL 2006*.
- Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of EMNLP*.
- Franz Joseph Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.
- Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL 2003*, pages 160–167.
- Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of IEEE*, pages 257–286.
- Ratnaparkhi and Adwait. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*.
- Stolcke and Andreas. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 311–318.
- Nianwen Xue and Libin Shen. 2003. Chinese word segmentation as lmr tagging. In *Proceedings of SIGHAN Workshop*.
- Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of ACL 2007*.

# Joint Processing and Discriminative Training for Letter-to-Phoneme Conversion

Sittichai Jiampojarn<sup>†</sup>

Colin Cherry<sup>‡</sup>

Grzegorz Kondrak<sup>†</sup>

<sup>†</sup>Department of Computing Science  
University of Alberta  
Edmonton, AB, T6G 2E8, Canada  
{sj, kondrak}@cs.ualberta.ca

<sup>‡</sup>Microsoft Research  
One Microsoft Way  
Redmond, WA, 98052

colinc@microsoft.com

## Abstract

We present a discriminative structure-prediction model for the letter-to-phoneme task, a crucial step in text-to-speech processing. Our method encompasses three tasks that have been previously handled separately: input segmentation, phoneme prediction, and sequence modeling. The key idea is online discriminative training, which updates parameters according to a comparison of the current system output to the desired output, allowing us to train all of our components together. By folding the three steps of a pipeline approach into a unified dynamic programming framework, we are able to achieve substantial performance gains. Our results surpass the current state-of-the-art on six publicly available data sets representing four different languages.

## 1 Introduction

Letter-to-phoneme (L2P) conversion is the task of predicting the pronunciation of a word, represented as a sequence of phonemes, from its orthographic form, represented as a sequence of letters. The L2P task plays a crucial role in speech synthesis systems (Schroeter et al., 2002), and is an important part of other applications, including spelling correction (Toutanova and Moore, 2001) and speech-to-speech machine translation (Engelbrecht and Schultz, 2005).

Converting a word into its phoneme representation is not a trivial task. Dictionary-based approaches cannot achieve this goal reliably, due to unseen words and proper names. Furthermore, the

construction of even a modestly-sized pronunciation dictionary requires substantial human effort for each new language. Effective rule-based approaches can be designed for some languages such as Spanish. However, Kominek and Black (2006) show that in languages with a less transparent relationship between spelling and pronunciation, such as English, Dutch, or German, the number of letter-to-sound rules grows almost linearly with the lexicon size. Therefore, most recent work in this area has focused on machine-learning approaches.

In this paper, we present a joint framework for letter-to-phoneme conversion, powered by online discriminative training. By updating our model parameters online, considering only the current system output and its feature representation, we are able to not only incorporate overlapping features, but also to use the same learning framework with increasingly complex search techniques. We investigate two online updates: averaged perceptron and Margin Infused Relaxed Algorithm (MIRA). We evaluate our system on L2P data sets covering English, French, Dutch and German. In all cases, our system outperforms the current state of the art, reducing the best observed error rate by as much as 46%.

## 2 Previous work

Letter-to-phoneme conversion is a complex task, for which a number of diverse solutions have been proposed. It is a structure prediction task; both the input and output are structured, consisting of sequences of letters and phonemes, respectively. This makes L2P a poor fit for many machine-learning techniques that are formulated for binary classification.

The L2P task is also characterized by the existence of a hidden structure connecting input to output. The training data consists of letter strings paired with phoneme strings, without explicit links connecting individual letters to phonemes. The subtask of inserting these links, called letter-to-phoneme alignment, is not always straightforward. For example, consider the word “phoenix” and its corresponding phoneme sequence [f i n i k s], where we encounter cases of two letters generating a single phoneme (ph→f), and a single letter generating two phonemes (x→k s). Fortunately, alignments between letters and phonemes can be discovered reliably with unsupervised generative models. Originally, L2P systems assumed one-to-one alignment (Black et al., 1998; Damper et al., 2005), but recently many-to-many alignment has been shown to perform better (Bisani and Ney, 2002; Jiampojarn et al., 2007). Given such an alignment, L2P can be viewed either as a sequence of classification problems, or as a sequence modeling problem.

In the classification approach, each phoneme is predicted independently using a multi-class classifier such as decision trees (Daelemans and Bosch, 1997; Black et al., 1998) or instance-based learning (Bosch and Daelemans, 1998). These systems predict a phoneme for each input letter, using the letter and its context as features. They leverage the structure of the input but ignore any structure in the output.

L2P can also be viewed as a sequence modeling, or tagging problem. These approaches model the structure of the output, allowing previously predicted phonemes to inform future decisions. The supervised Hidden Markov Model (HMM) applied by Taylor (2005) achieved poor results, mostly because its maximum-likelihood emission probabilities cannot be informed by the emitted letter’s context. Other approaches, such as those of Bisani and Ney (2002) and Marchand and Damper (2000), have shown that better performance can be achieved by pairing letter substrings with phoneme substrings, allowing context to be captured implicitly by these groupings.

Recently, two hybrid methods have attempted to capture the flexible context handling of classification-based methods, while also modeling the sequential nature of the output. The

constraint satisfaction inference (CSInf) approach (Bosch and Canisius, 2006) improves the performance of instance-based classification (Bosch and Daelemans, 1998) by predicting for each letter a trigram of phonemes consisting of the previous, current and next phonemes in the sequence. The final output sequence is the sequence of predicted phonemes that satisfies the most unigram, bigram and trigram agreement constraints. The second hybrid approach (Jiampojarn et al., 2007) also extends instance-based classification. It employs a many-to-many letter-to-phoneme alignment model, allowing substrings of letters to be classified into substrings of phonemes, and introducing an input segmentation step before prediction begins. The method accounts for sequence information with post-processing: the numerical scores of possible outputs from an instance-based phoneme predictor are combined with phoneme transition probabilities in order to identify the most likely phoneme sequence.

### 3 A joint approach

By observing the strengths and weaknesses of previous approaches, we can create the following prioritized desiderata for any L2P system:

1. The phoneme predicted for a letter should be informed by the letter’s context in the input word.
2. In addition to single letters, letter substrings should also be able to generate phonemes.
3. Phoneme sequence information should be included in the model.

Each of the previous approaches focuses on one or more of these items. Classification-based approaches such as the decision tree system (Black et al., 1998) and instance-based learning system (Bosch and Daelemans, 1998) take into account the letter’s context (#1). By pairing letter substrings with phoneme substrings, the joint n-gram approach (Bisani and Ney, 2002) accounts for all three desiderata, but each operation is informed only by a limited amount of left context. The many-to-many classifier of Jiampojarn et al. (2007) also attempts to account for all three, but it adheres

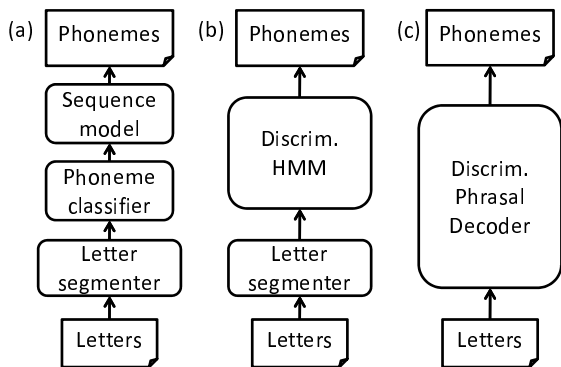


Figure 1: Collapsing the pipeline.

strictly to the pipeline approach illustrated in Figure 1a. It applies in succession three separately trained modules for input segmentation, phoneme prediction, and sequence modeling. Similarly, the CSInf approach modifies independent phoneme predictions (#1) in order to assemble them into a cohesive sequence (#3) in post-processing.

The pipeline approaches are undesirable for two reasons. First, when decisions are made in sequence, errors made early in the sequence can propagate forward and throw off later processing. Second, each module is trained independently, and the training methods are not aware of the tasks performed later in the pipeline. For example, optimal parameters for a phoneme prediction module may vary depending on whether or not the module will be used in conjunction with a phoneme sequence model.

We propose a joint approach to L2P conversion, grounded in dynamic programming and online discriminative training. We view L2P as a tagging task that can be performed with a discriminative learning method, such as the Perceptron HMM (Collins, 2002). The Perceptron HMM naturally handles phoneme prediction (#1) and sequence modeling (#3) simultaneously, as shown in Figure 1b. Furthermore, unlike a generative HMM, it can incorporate many overlapping source  $n$ -gram features to represent context. In order to complete the conversion from a pipeline approach to a joint approach, we fold our input segmentation step into the exact search framework by replacing a separate segmentation module (#2) with a monotone phrasal decoder (Zens and Ney, 2004). At this point all three of our desiderata are incorporated into a single module,

---

**Algorithm 1** Online discriminative training.

---

```

1:  $\alpha = \vec{0}$ 
2: for  $K$  iterations over training set do
3:   for all letter-phoneme sequence pairs  $(x, y)$ 
     in the training set do
4:      $\hat{y} = \arg \max_{y' \in Y} [\alpha \cdot \Phi(x, y')]$ 
5:     update weights  $\alpha$  according to  $\hat{y}$  and  $y$ 
6:   end for
7: end for
8: return  $\alpha$ 

```

---

as shown in Figure 1c.

Our joint approach to L2P lends itself to several refinements. We address an underfitting problem of the perceptron by replacing it with a more robust Margin Infused Relaxed Algorithm (MIRA), which adds an explicit notion of margin and takes into account the system’s current  $n$ -best outputs. In addition, with all of our features collected under a unified framework, we are free to conjoin context features with sequence features to create a powerful linear-chain model (Sutton and McCallum, 2006).

## 4 Online discriminative training

In this section, we describe our entire L2P system. An outline of our discriminative training process is presented in Algorithm 1. An online process repeatedly finds the best output(s) given the current weights, and then updates those weights to make the model favor the correct answer over the incorrect ones.

The system consists of the following three main components, which we describe in detail in Sections 4.1, 4.2 and 4.3, respectively.

1. A scoring model, represented by a weighted linear combination of features ( $\alpha \cdot \Phi(x, y)$ ).
2. A search for the highest scoring phoneme sequence for a given input word (Step 4).
3. An online update equation to move the model away from incorrect outputs and toward the correct output (Step 5).

### 4.1 Model

Given an input word  $x$  and an output phoneme sequence  $y$ , we define  $\Phi(x, y)$  to be a feature vector

representing the evidence for the sequence  $y$  found in  $x$ , and  $\alpha$  to be a feature weight vector providing a weight for each component of  $\Phi(x, y)$ . We assume that both the input and output consist of  $m$  substrings, such that  $x_i$  generates  $y_i$ ,  $0 \leq i < m$ . At training time, these substrings are taken from a many-to-many letter-to-phoneme alignment. At test time, input segmentation is handled by either a segmentation module or a phrasal decoder.

Table 1 shows our feature template that we include in  $\Phi(x, y)$ . We use only indicator features; each feature takes on a binary value indicating whether or not it is present in the current  $(x, y)$  pair. The context features express letter evidence found in the input string  $x$ , centered around the generator  $x_i$  of each  $y_i$ . The parameter  $c$  establishes the size of the context window. Note that we consider not only letter unigrams but all  $n$ -grams that fit within the window, which enables the model to assign phoneme preferences to contexts containing specific sequences, such as *ing* and *tion*. The transition features are HMM-like sequence features, which enforce cohesion on the output side. We include only first-order transition features, which look back to the previous phoneme substring generated by the system, because our early development experiments indicated that larger histories had little impact on performance; however, the number of previous substrings that are taken into account could be extended at a polynomial cost. Finally, the linear-chain features (Sutton and McCallum, 2006) associate the phoneme transitions between  $y_{i-1}$  and  $y_i$  with each  $n$ -gram surrounding  $x_i$ . This combination of sequence and context data provides the model with an additional degree of control.

## 4.2 Search

Given the current feature weight vector  $\alpha$ , we are interested in finding the highest-scoring phoneme sequence  $\hat{y}$  in the set  $Y$  of all possible phoneme sequences. In the pipeline approach (Figure 1b), the input word is segmented into letter substrings by an instance-based classifier (Aha et al., 1991), which learns a letter segmentation model from many-to-many alignments (Jiampoamarn et al., 2007). The search for the best output sequence is then effectively a substring tagging problem, and we can compute the  $\arg \max$  operation in line 4 of Algorithm 1

context	$x_{i-c}, y_i$
	...
	$x_{i+c}, y_i$
	$x_{i-c}x_{i-c+1}, y_i$
	...
	$x_{i+c-1}x_{i+c}, y_i$
.....	
	$x_{i-c} \dots x_{i+c}, y_i$
transition	$y_{i-1}, y_i$
linear chain	$x_{i-c}, y_{i-1}, y_i$
	...
	$x_{i+c}, y_{i-1}, y_i$
	$x_{i-c}x_{i-c+1}, y_{i-1}, y_i$
	...
	$x_{i+c-1}x_{i+c}, y_{i-1}, y_i$
	.....
	$x_{i-c} \dots x_{i+c}, y_{i-1}, y_i$

Table 1: Feature template.

with the standard HMM Viterbi search algorithm.

In the joint approach (Figure 1c), we perform segmentation and L2P prediction simultaneously by applying the monotone search algorithm developed for statistical machine translation (Zens and Ney, 2004). Thanks to its ability to translate phrases (in our case, letter substrings), we can accomplish the  $\arg \max$  operation without specifying an input segmentation in advance; the search enumerates all possible segmentations. Furthermore, the language model functionality of the decoder allows us to keep benefiting from the transition and linear-chain features, which are explicit in the previous HMM approach.

The search can be efficiently performed by the dynamic programming recurrence shown below. We define  $Q(j, p)$  as the maximum score of the phoneme sequence ending with the phoneme  $p$  generated by the letter sequence  $x_1 \dots x_j$ . Since we are no longer provided an input segmentation in advance, in this framework we view  $x$  as a sequence of  $J$  letters, as opposed to substrings. The phoneme  $p'$  is the phoneme produced in the previous step. The expression  $\phi(x_{j'+1}^j, p', p)$  is a convenient way to express the subvector of our complete feature vector  $\Phi(x, y)$  that describes the substring pair  $(x_i, y_{i-1}^i)$ , where  $x_i = x_{j'+1}^j$ ,  $y_{i-1} = p'$  and  $y_i = p$ . The value  $N$  limits the size of the dynamically created



substrings. We use  $N = 2$ , which reflects a similar limit in our many-to-many aligner. The special symbol  $\$$  represents a starting phoneme or ending phoneme. The value in  $Q(I + 1, \$)$  is the score of highest scoring phoneme sequence corresponding to the input word. The actual sequence can be retrieved by backtracking through the table  $Q$ .

$$Q(0, \$) = 0$$

$$Q(j, p) = \max_{\substack{p', p, \\ j-N \leq j' < j}} \{\alpha \cdot \phi(x_{j'+1}^j, p', p) + Q(j', p')\}$$

$$Q(J + 1, \$) = \max_{p'} \{\alpha \cdot \phi(\$, p', \$) + Q(J, p')\}$$

### 4.3 Online update

We investigate two model updates to drive our online discriminative learning. The simple perceptron update requires only the system’s current output, while MIRA allows us to take advantage of the system’s current  $n$ -best outputs.

#### Perceptron

Learning a discriminative structure prediction model with a perceptron update was first proposed by Collins (2002). The perceptron update process is relatively simple, involving only vector addition. In line 5 of Algorithm 1, the weight vector  $\alpha$  is updated according to the best output  $\hat{y}$  under the current weights and the true output  $y$  in the training data. If  $\hat{y} = y$ , there is no update to the weights; otherwise, the weights are updated as follows:

$$\alpha = \alpha + \Phi(x, y) - \Phi(x, \hat{y}) \quad (1)$$

We iterate through the training data until the system performance drops on a held-out set. In a separable case, the perceptron will find an  $\alpha$  such that:

$$\forall \hat{y} \in Y - \{y\} : \alpha \cdot \Phi(x, y) > \alpha \cdot \Phi(x, \hat{y}) \quad (2)$$

Since real-world data is not often separable, the average of all  $\alpha$  values seen throughout training is used in place of the final  $\alpha$ , as the average generalizes better to unseen data.

#### MIRA

In the perceptron training algorithm, no update is derived from a particular training example so long as the system is predicting the correct phoneme sequence. The perceptron has no notion of margin: a

slim preference for the correct sequence is just as good as a clear preference. During development, we observed that this lead to underfitting the training examples; useful and consistent evidence was ignored because of the presence of stronger evidence in the same example. The MIRA update provides a principled method to resolve this problem.

The Margin Infused Relaxed Algorithm or MIRA (Crammer and Singer, 2003) updates the model based on the system’s  $n$ -best output. It employs a margin update which can induce an update even when the 1-best answer is correct. It does so by finding a weight vector that separates incorrect sequences in the  $n$ -best list from the correct sequence by a variable width margin.

The update process finds the smallest change in the current weights so that the new weights will separate the correct answer from each incorrect answer by a margin determined by a structured loss function. The loss function describes the distance between an incorrect prediction and the correct one; that is, it quantifies just how wrong the proposed sequence is. This update process can be described as an optimization problem:

$$\begin{aligned} \min_{\alpha_n} \quad & \|\alpha_n - \alpha_o\| \\ \text{subject to } & \forall \hat{y} \in Y_n : \\ & \alpha_n \cdot (\Phi(x, y) - \Phi(x, \hat{y})) \geq \ell(y, \hat{y}) \end{aligned} \quad (3)$$

where  $Y_n$  is a set of  $n$ -best outputs found under the current model,  $y$  is the correct answer,  $\alpha_o$  is the current weight vector,  $\alpha_n$  is the new weight vector, and  $\ell(y, \hat{y})$  is the loss function.

Since our direct objective is to produce the correct phoneme sequence for a given word, the most intuitive way to define the loss function  $\ell(y, \hat{y})$  is binary: 0 if  $\hat{y} = y$ , and 1 otherwise. We refer to this as 0-1 loss. Another possibility is to base the loss function on the phoneme error rate, calculated as the Levenshtein distance between  $y$  and  $\hat{y}$ . We can also compute a combined loss function as an equally-weighted linear combination of the 0-1 and phoneme loss functions.

MIRA training is similar to averaged perceptron training, but instead of finding the single best answer, we find the  $n$ -best answers ( $Y_n$ ) and update weights according to Equation 3. To find the  $n$ -best answers, we modify the HMM and monotone search algorithms to keep track of the  $n$ -best phonemes at

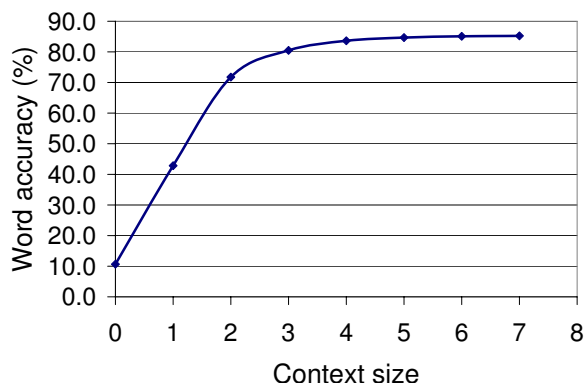


Figure 2: Perceptron update with different context size.

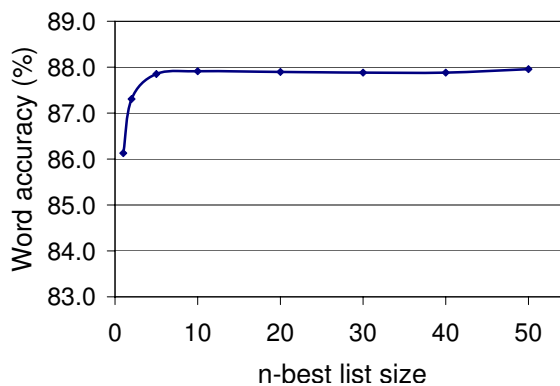


Figure 3: MIRA update with different size of  $n$ -best list.

each cell of the dynamic programming matrix. The optimization in Equation 3 is a standard quadratic programming problem that can be solved by using Hildreth’s algorithm (Censor and Zenios, 1997). The details of our implementation of MIRA within the SVM<sup>light</sup> framework (Joachims, 1999) are given in the Appendix A. Like the perceptron algorithm, MIRA returns the average of all weight vectors produced during learning.

## 5 Evaluation

We evaluated our approach on English, German and Dutch CELEX (Baayen et al., 1996), French Brulex, English Nttalk and English CMUDict data sets. Except for English CELEX, we used the data sets from the PRONALSYL letter-to-phoneme conversion challenge<sup>1</sup>. Each data set is divided into 10 folds: we used the first one for testing, and the rest for training. In all cases, we hold out 5% of our training data to determine when to stop perceptron or MIRA training. We ignored one-to-one alignments included in the PRONALSYL data sets, and instead induced many-to-many alignments using the method of Jiampoamarn et al. (2007).

Our English CELEX data set was extracted directly from the CELEX database. After removing duplicate words, phrases, and abbreviations, the data set contained 66,189 word-phoneme pairs, of which 10% was designated as the final test set, and the rest as the training set. We performed our development experiments on the latter part, and then used the final

test set to compare the performance of our system to other results reported in the literature.

We report the system performance in terms of word accuracy, which rewards only completely correct phoneme sequences. Word accuracy is more demanding than phoneme accuracy, which considers the number of correct phonemes. We feel that word accuracy is a more appropriate error metric, given the quality of current L2P systems. Phoneme accuracy is not sensitive enough to detect improvements in highly accurate L2P systems: Black et al. (1998) report 90% phoneme accuracy is equivalent to approximately 60% word accuracy, while 99% phoneme accuracy corresponds to only 90% word accuracy.

### 5.1 Development Experiments

We began development with a zero-order Perceptron HMM with an external segmenter, which uses only the context features from Table 1. The zero-order Perceptron HMM is equivalent to training a multi-class perceptron to make independent substring-to-phoneme predictions; however, this framework allows us to easily extend to structured models. We investigate the effect of augmenting this baseline system in turn with larger context sizes, the MIRA update, joint segmentation, and finally sequence features. We report the impact of each contribution on our English CELEX development set.

Figure 2 shows the performance of our baseline L2P system with different context size values ( $c$ ). Increasing the context size has a dramatic effect on accuracy, but the effect begins to level off for context sizes greater than 5. Henceforth, we report the

<sup>1</sup>Available at <http://www.pascal-network.org/Challenges/PRONALSYL/>. The results have not been announced.

	Perceptron	MIRA
Separate segmentation	84.5%	85.8%
Phrasal decoding	<b>86.6%</b>	<b>88.0%</b>

Table 2: Separate segmentation versus phrasal decoding in terms of word accuracy.

results with context size  $c = 5$ .

Figure 3 illustrates the effect of varying the size of  $n$ -best list in the MIRA update.  $n = 1$  is equivalent to taking into account only the best answer, which does not address the underfitting problem. A large  $n$ -best list makes it difficult for the optimizer to separate the correct and incorrect answers, resulting in large updates at each step. We settle on  $n = 10$  for the subsequent experiments.

The choice of MIRA’s loss function has a minimal impact on performance, probably because our baseline system already has a very high phoneme accuracy. We employ the loss function that combines  $0-1$  and phoneme error rate, due to its marginal improvement over  $0-1$  loss on the development set.

Looking across columns in Table 2, we observe over 8% reduction in word error rate when the perceptron update is replaced with the MIRA update. Since the perceptron is a considerably simpler algorithm, we continue to report the results of both variants throughout this section.

Table 2 also shows the word accuracy of our system after adding the option to conduct joint segmentation through phrasal decoding. The 15% relative reduction in error rate in the second row demonstrates the utility of folding the segmentation step into the search. It also shows that the joint framework enables the system to reduce and compensate for errors that occur in a pipeline. This is particularly interesting because our separate instance-based segmenter is highly accurate, achieving 98% segmentation accuracy. Our experiments indicate that the application of joint segmentation recovers more than 60% of the available improvements, according to an upper bound determined by utilizing perfect segmentation.<sup>2</sup>

Table 3 illustrates the effect of our sequence features on both the perceptron and MIRA systems.

<sup>2</sup>Perfect with respect to our many-to-many alignment (Jiampojarn et al., 2007), but not necessarily in any linguistic sense.

Feature	Perceptron	MIRA
zero order	86.6%	88.0%
+ 1 <sup>st</sup> order HMM	87.1%	88.3%
+ linear-chain	87.5%	89.3%
All features	<b>87.8%</b>	<b>89.4%</b>

Table 3: The effect of sequence features on the joint system in terms of word accuracy.

Replacing the zero-order HMM with the first-order HMM makes little difference by itself, but combined with the more powerful linear-chain features, it results in a relative error reduction of about 12%. In general, the linear-chain features make a much larger difference than the relatively simple transition features, which underscores the importance of using source-side context when assessing sequences of phonemes.

The results reported in Tables 2 and 3 were calculated using cross validation on the training part of the CELEX data set. With the exception of adding the 1<sup>st</sup> order HMM, the differences between versions are statistically significant according to McNemar’s test at 95% confidence level. On one CPU of AMD Opteron 2.2GHz with 6GB of installed memory, it takes approximately 32 hours to train the MIRA model with all features, compared to 12 hours for the zero-order model.

## 5.2 System Comparison

Table 4 shows the comparison between our approach and other systems on the evaluation data sets. We trained our system using  $n$ -gram context, transition, and linear-chain features. All parameters, including the size of  $n$ -best list, size of letter context, and the choice of loss functions, were established on the English CELEX development set, as presented in our previous experiments. With the exception of the system described in (Jiampojarn et al., 2007), which we re-ran on our current test sets, the results of other systems are taken from the original papers. Although these comparisons are necessarily indirect due to different experimental settings, they strongly suggest that our system outperforms all previous published results on all data sets, in some case by large margins. When compared to the current state-of-the-art performance of each data set, the relative reductions in error rate range from 7% to 46%.

Corpus	MIRA	Perceptron	M-M HMM	Joint n-gram*	CSInf*	PbA*	CART*
Eng. CELEX	<b>90.51%</b>	88.44%	<u>84.81%</u>	76.3%	84.5%	-	-
Dutch CELEX	<b>95.32%</b>	95.13%	91.69%	-	<u>94.5%</u>	-	-
German CELEX	<b>93.61%</b>	92.84%	90.31%	<u>92.5%</u>	-	-	89.38%
Nettalk	<b>67.82%</b>	64.87%	59.32%	64.6%	-	<u>65.35%</u>	-
CMUDict	<b>71.99%</b>	71.03%	<u>65.38%</u>	-	-	-	57.80%
Brulex	<b>94.51%</b>	93.89%	<u>89.77%</u>	89.1%	-	-	-

Table 4: Word accuracy on the evaluated data sets. **MIRA**, **Perceptron**: our systems. **M-M HMM**: Many-to-Many HMM system (Jiampojamarn et al., 2007). **Joint n-gram**: Joint n-gram model (Demberg et al., 2007). **CSInf**: Constraint satisfaction inference (Bosch and Canisius, 2006). **PbA**: Pronunciation by Analogy (Marchand and Dampier, 2006). **CART**: CART decision tree system (Black et al., 1998). The columns marked with \* contain results reported in the literature. “-” indicates no reported results. We have underlined the best previously reported results.

## 6 Conclusion

We have presented a joint framework for letter-to-phoneme conversion, powered by online discriminative training. We introduced two methods to convert multi-letter substrings into phonemes: one relying on a separate segmenter, and the other incorporating a unified search that finds the best input segmentation while generating the output sequence. We investigated two online update algorithms: the perceptron, which is straightforward to implement, and MIRA, which boosts performance by avoiding underfitting. Our systems employ source  $n$ -gram features and linear-chain features, which substantially increase L2P accuracy. Our experimental results demonstrate the power of a joint approach based on online discriminative training with large feature sets. In all cases, our MIRA-based system advances the current state of the art by reducing the best reported error rate.

### Appendix A. MIRA Implementation

We optimize the objective shown in Equation 3 using the SVM<sup>light</sup> framework (Joachims, 1999), which provides the quadratic program solver shown in Equation 4.

$$\begin{aligned}
 & \min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\
 & \text{subject to } \forall i, \\
 & w \cdot t_i \geq rhs_i - \xi_i
 \end{aligned} \tag{4}$$

In order to approximate a hard margin using the soft-margin optimizer of SVM<sup>light</sup>, we assign a very large penalty value to  $C$ , thus making the use of any slack variables ( $\xi_i$ ) prohibitively expensive. We define the vector  $w$  as the difference between the new

and previous weights:  $w = \alpha_n - \alpha_o$ . We constrain  $w$  to mirror the constraints in Equation 3. Since each  $\hat{y}$  in the  $n$ -best list ( $Y_n$ ) needs a constraint based on its feature difference vector, we define a  $t_i$  for each:

$$\forall \hat{y} \in Y_n : t_i = \Phi(x, y) - \Phi(x, \hat{y})$$

Substituting that equation along with the inferred equation  $a_n = a_o + w$  into our original MIRA constraints yields:

$$(\alpha_o + w) \cdot t_i \geq \ell(y, \hat{y})$$

Moving  $\alpha_o$  to the right-hand-side to isolate  $w \cdot t_i$  on the left, we get a set of mappings that implement MIRA in SVM<sup>light</sup>'s optimizer:

$w$	$\alpha_n - \alpha_o$
$t_i$	$\Phi(x, y) - \Phi(x, \hat{y})$
$rhs_i$	$\ell(y, \hat{y}) - \alpha_o \cdot t_i$

The output of the SVM<sup>light</sup> optimizer is an update vector  $w$  to be added to the current  $\alpha_o$ .

### Acknowledgements

This research was supported by the Alberta Ingenuity Fund, and the Natural Sciences and Engineering Research Council of Canada.

### References

- David W. Aha, Dennis Kibler, and Marc K. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.
- Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1996. The CELEX2 lexical database. LDC96L14.

- Maximilian Bisani and Hermann Ney. 2002. Investigations on joint-multigram models for grapheme-to-phoneme conversion. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 105–108.
- Alan W. Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The Third ESCA Workshop in Speech Synthesis*, pages 77–80.
- Antal Van Den Bosch and Sander Canisius. 2006. Improved morpho-phonological sequence processing with constraint satisfaction inference. *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology, SIGPHON '06*, pages 41–49.
- Antal Van Den Bosch and Walter Daelemans. 1998. Do not forget: Full memory in memory-based learning of word pronunciation. In *Proceedings of NeM-LaP3/CoNLL98*, pages 195–204, Sydney, Australia.
- Yair Censor and Stavros A. Zenios. 1997. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press.
- Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, Morristown, NJ, USA.
- Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991.
- Walter Daelemans and Antal Van Den Bosch. 1997. Language-independent data-oriented grapheme-to-phoneme conversion. In *Progress in Speech Synthesis*, pages 77–89. New York, USA.
- Robert I. Damber, Yannick Marchand, John DS. Marsters, and Alexander I. Bazin. 2005. Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology*, 8(2):147–160.
- Vera Demberg, Helmut Schmid, and Gregor Möhler. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 96–103, Prague, Czech Republic.
- Herman Engelbrecht and Tanja Schultz. 2005. Rapid development of an afrikaans-english speech-to-speech translator. In *International Workshop of Spoken Language Translation (IWSLT)*, Pittsburgh, PA, USA.
- Sittichai Jiampojarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York, USA.
- Thorsten Joachims. 1999. Making large-scale support vector machine learning practical. pages 169–184. MIT Press, Cambridge, MA, USA.
- John Kominek and Alan W Black. 2006. Learning pronunciation dictionaries: Language complexity and word selection strategies. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 232–239, New York City, USA.
- Yannick Marchand and Robert I. Damber. 2000. A multistrategy approach to improving pronunciation by analogy. *Computational Linguistics*, 26(2):195–219.
- Yannick Marchand and Robert I. Damber. 2006. Can syllabification improve pronunciation by analogy of English? *Natural Language Engineering*, 13(1):1–24.
- Juergen Schroeter, Alistair Conkie, Ann Syrdal, Mark Beutnagel, Matthias Jilka, Volker Strom, Yeon-Jun Kim, Hong-Goo Kang, and David Kapilow. 2002. A perspective on the next challenges for TTS research. In *IEEE 2002 Workshop on Speech Synthesis*.
- Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.
- Paul Taylor. 2005. Hidden Markov Models for grapheme to phoneme conversion. In *Proceedings of the 9th European Conference on Speech Communication and Technology*.
- Kristina Toutanova and Robert C. Moore. 2001. Pronunciation modeling for improved spelling correction. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151, Morristown, NJ, USA.
- Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 257–264, Boston, Massachusetts, USA.

# A Probabilistic Model for Fine-Grained Expert Search

Shenghua Bao<sup>1</sup>, Huizhong Duan<sup>1</sup>, Qi Zhou<sup>1</sup>, Miao Xiong<sup>1</sup>, Yunbo Cao<sup>1,2</sup>, Yong Yu<sup>1</sup>

<sup>1</sup>Shanghai Jiao Tong University,  
Shanghai, China, 200240

{shhbao, summer, jackson, xiongmiao, yyu}  
@apex.sjtu.edu.cn

<sup>2</sup>Microsoft Research Asia  
Beijing, China, 100080

yunbo.cao@microsoft.com

## Abstract

Expert search, in which given a query a ranked list of experts instead of documents is returned, has been intensively studied recently due to its importance in facilitating the needs of both information access and knowledge discovery. Many approaches have been proposed, including metadata extraction, expert profile building, and formal model generation. However, all of them conduct expert search with a coarse-grained approach. With these, further improvements on expert search are hard to achieve. In this paper, we propose conducting expert search with a fine-grained approach. Specifically, we utilize more specific evidences existing in the documents. An evidence-oriented probabilistic model for expert search and a method for the implementation are proposed. Experimental results show that the proposed model and the implementation are highly effective.

## 1 Introduction

Nowadays, team work plays a more important role than ever in problem solving. For instance, within an enterprise, people handle new problems usually by leveraging the knowledge of experienced colleagues. Similarly, within research communities, novices step into a new research area often by learning from well-established researchers in the research area. All these scenarios involve asking the questions like “who is an expert on X?” or “who knows about X?” Such questions, which cannot be answered easily through traditional document search, raise a new requirement of searching people with certain expertise.

To meet that requirement, a new task, called *expert search*, has been proposed and studied intensively. For example, TREC 2005, 2006, and 2007

provide the task of *expert search* within the enterprise track. In the TREC setting, *expert search* is defined as: given a query, a ranked list of experts is returned. In this paper, we engage our study in the same setting.

Many approaches to expert search have been proposed by the participants of TREC and other researchers. These approaches include metadata extraction (Cao et al., 2005), expert profile building (Craswell, 2001, Fu et al., 2007), data fusion (Macdonald and Ounis, 2006), query expansion (Macdonald and Ounis, 2007), hierarchical language model (Petkova and Croft, 2006), and formal model generation (Balog et al., 2006; Fang et al., 2006). However, all of them conduct expert search with what we call a coarse-grained approach. The discovering and use of evidence for expert locating is carried out under a grain of document. With it, further improvements on expert search are hard to achieve. This is because different blocks (or segments) of electronic documents usually present different functions and qualities and thus different impacts for expert locating.

In contrast, this paper is concerned with proposing a probabilistic model for fine-grained expert search. In fine-grained expert search, we are to extract and use evidence of expert search (usually blocks of documents) directly. Thus, the proposed probabilistic model incorporates evidence of expert search explicitly as a part of it. A piece of *fine-grained evidence* is formally defined as a quadruple,  $\langle \textit{topic}, \textit{person}, \textit{relation}, \textit{document} \rangle$ , which denotes the fact that a *topic* and a *person*, with a certain *relation* between them, are found in a specific *document*. The intuition behind the quadruple is that a query may be matched with phrases in various forms (denoted as *topic* here) and an *expert candidate* may appear with various name masks (denoted as *person* here), e.g., full name, email, or abbreviated names. Given a *topic* and *person, relation* type is used to measure their closeness and

*document* serves as a context indicating whether it is good *evidence*.

Our proposed model for fine-grained expert search results in an implementation of two stages.

1) Evidence Extraction: document segments in various granularities are identified and evidences are extracted from them. For example, we can have segments in which an expert candidate and a queried topic co-occur within a same section of *document-001*: "...later, *Berners-Lee* describes a *semantic web search engine* experience..." As the result, we can extract an evidence by using same-section relation, i.e., *<semantic web search engine, Berners-Lee, same-section, document-001>*.

2) Evidence Quality Evaluation: the quality (or reliability) of evidence is evaluated. The quality of a quadruple of evidence consists of four aspects, namely *topic-matching quality*, *person-name-matching quality*, *relation quality*, and *document quality*. If we regard evidence as link of expert candidate and queried topic, the four aspects will correspond to the strength of the link to query, the strength of the link to expert candidate, the type of the link, and the document context of the link respectively.

All the *evidences* with their scores of *quality* are merged together to generate a single score for each expert candidate with regard to a given query. We empirically evaluate our proposed model and implementation on the W3C corpus which is used in the expert search task at TREC 2005 and 2006. Experimental results show that both explored evidences and evaluation of evidence quality can improve the expert search significantly. Compared with existing state-of-the-art expert search methods, the probabilistic model for fine-grained expert search shows promising improvement.

The rest of the paper is organized as follows. Section 2 surveys existing studies on expert search. Section 3 and Section 4 present the proposed probabilistic model and its implementation, respectively. Section 5 gives the empirical evaluation. Finally, Section 6 concludes the work.

## 2 Related Work

### 2.1 Expert Search Systems

One setting for automatic expert search is to assume that data from specific resources are available. For example, Expertise Recommender (Kautz

et al., 1996), Expertise Browser (Mockus and Herbsleb, 2002) and the system in (McDonald and Ackerman, 1998) make use of log data in software development systems to find experts. Yet another approach is to mine expert and expertise from email communications (Campbell et al., 2003; Dom et al. 2003; Sihn and Heeren, 2001).

Searching expert from general documents has also been studied (Davenport and Prusak, 1998; Mattox et al., 1999; Hertzum and Pejtersen, 2000). P@NOPTIC employs what is referred to as the 'profile-based' approach in searching for experts (Craswell et al., 2001). Expert/Expert-Locating (EEL) system (Steer and Lochbaum, 1988) uses the same approach in searching for expert groups. DEMOIR (Yimam, 1996) enhances the profile-based approach by separating co-occurrences into different types. In essence, the profile-based approach utilizes the co-occurrences between query words and people within documents.

### 2.2 Expert Search at TREC

A task on expert search was organized within the enterprise track at TREC 2005, 2006 and 2007 (Craswell et al., 2005; Soboroff et al., 2006; Bailey et al., 2007).

Many approaches have been proposed for tackling the expert search task within the TREC track. Cao et al. (2005) propose a two-stage model with a set of extracted metadata. Balog et al. (2006) compare two generative models for expert search. Fang et al. (2006) further extend their generative model by introducing the prior of expert distribution and relevance feedback. Petkova and Croft (2006) further extend the profile based method by using a hierarchical language model. Macdonald and Ounis (2006) investigate the effectiveness of the voting approach and the associated data fusion techniques. However, such models are conducted in a coarse-grain scope of document as discussed before. In contrast, our study focuses on proposing a model for conducting expert search in a fine-grain scope of evidence (local context).

## 3 Fine-grained Expert Search

Our research is to investigate a direct use of the local contexts for expert search. We call each local context of such kind as *fine-grained evidence*.

In this work, a fine-grained evidence is formally defined as a quadruple, *<topic, person, relation,*

*document*>. Such a quadruple denotes that a *topic* and a *person* occurrence, with a certain *relation* between them, are found in a specific *document*.

Recall that *topic* is different from *query*. For example, given a query “*semantic web coordination*”, the corresponding *topic* may be either “*semantic web*” or “*web coordination*”. Similarly, *person* here is different from *expert candidate*. E.g, given an *expert candidate* “*Ritu Raj Tiwari*”, the matched *person* may be “*Ritu Raj Tiwari*”, “*Tiwari*”, or “*RRT*” etc. Although both the topics and persons may not match the query and expert candidate exactly, they do have certain indication on the connection of query “*semantic web coordination*” and expert “*Ritu Raj Tiwari*”.

### 3.1 Evidence-Oriented Expert Search Model

We conduct fine-grained expert search by incorporating evidence of local context *explicitly* in a probabilistic model which we call an *evidence-oriented expert search model*. Given a query  $q$ , the probability of a candidate  $c$  being an expert (or knowing something about  $q$ ) is estimated as

$$\begin{aligned} P(c|q) &= \sum_e P(c, e|q) \\ &= \sum_e P(c|e, q)P(e|q) \end{aligned} \quad (1)$$

where  $e$  denotes a quadruple of *evidence*.

Using the relaxation that the probability of  $c$  is independent of a query  $q$  given an evidence  $e$ , we can reduce Equation (1) as,

$$P(c|q) = \sum_e P(c|e)P(e|q). \quad (2)$$

Compared to previous work, our model conducts expert search with a new way in which local contexts of *evidence* are used to bridge a query  $q$  and an expert candidate  $c$ . The new way enables the expert search system to explore various local contexts in a precise manner.

In the following sub-sections, we will detail two sub-models: the *expert matching model*  $P(c|e)$  and the *evidence matching model*  $P(e|q)$ .

### 3.2 Expert Matching Model

We expand the evidence  $e$  as quadruple  $\langle \textit{topic}, \textit{people}, \textit{relation}, \textit{document} \rangle$  ( $\langle t, p, r, d \rangle$  for short) for expert matching. Given a set of related evidences, we assume that the generation of an expert candidate  $c$  is independent with topic  $t$  and omit it

in expert matching. Therefore, we simplify the expert matching formula as below:

$$P(c|e) = P(c|p, r, d) = P(c|p)P(p|r, d), \quad (3)$$

where  $P(c|p)$  depends on how an expert candidate  $c$  matches to a person occurrence  $p$  (e.g. full name or email of a person). The different ways of matching an expert candidate  $c$  with a person occurrence  $p$  results in varied qualities.  $P(c|p)$  represents the quality.  $P(p|r, d)$  expresses the probability of an occurrence  $p$  given a relation  $r$  and a document  $d$ .  $P(p|r, d)$  is estimated in MLE as,

$$P(p|r, d) = \frac{\textit{freq}(p, r, d)}{L(r, d)}, \quad (4)$$

where  $\textit{freq}(p, r, d)$  is the frequency of person  $p$  matched by relation  $r$  in document  $d$ , and  $L(r, d)$  is the frequency of all the persons matched by relation  $r$  in  $d$ . This estimation can further be smoothed by using the evidence collection as follows:

$$P_3(p|r, d) = \mu P(p|r, d) + (1 - \mu) \sum_{d' \in D} \frac{P(p|r, d')}{|D|}, \quad (5)$$

where  $D$  denotes the whole document collection.  $|D|$  is the total number of documents.

We use Dirichlet prior in smoothing of parameter  $\mu$ :

$$\mu = \frac{L(r, d)}{L(r, d) + K}, \quad (6)$$

where  $K$  is the average frequency of all the experts in the collection.

### 3.3 Evidence Matching Model

By expanding the evidence  $e$  and employing independence assumption, we have the following formula for evidence matching:

$$\begin{aligned} P(e|q) &= P(t, p, r, d|q) \\ &= P(t|q)P(p|q)P(r|q)P(d|q) \end{aligned} \quad (7)$$

In the following, we are to explain what these four terms represent and how they can be estimated.

The first term  $P(t|q)$  represents the probability that a query  $q$  matches to a topic  $t$  in evidence. Recall that a query  $q$  may match a topic  $t$  in various ways, not necessarily being identical to  $t$ . For example, both topic “*semantic web*” and “*semantic web search engine*” can match the query “*semantic web search engine*”. The probability is defined as



$$P(t|q) \propto P(\text{type}(t,q)), \quad (8)$$

where  $\text{type}(t, q)$  represents the way that  $q$  matches to  $t$ , e.g., *phrase matching*. Different matching methods are associated with different probabilities.

The second term  $P(p|q)$  represents the probability that a person  $p$  is generated from a query  $q$ . The probability is further approximated by the prior probability of  $p$ ,

$$P(p|q) \propto P(p). \quad (9)$$

The prior probability can be estimated by MLE, i.e., the ratio of total occurrences of person  $p$  in the collection.

The third term represents the probability that a relation  $r$  is generated from a query  $q$ . Here, we approximate the probability as

$$P(r|q) \propto P(\text{type}(r)), \quad (10)$$

where  $\text{type}(r)$  represents the way  $r$  connecting query and expert.  $P(\text{type}(r))$  represents the reliability of relation type of  $r$ .

Following the Bayes rule, the last term can be transformed as

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \propto P(q|d)P(d), \quad (11)$$

where priority distribution  $P(d)$  can be estimated based on static rank, e.g., PageRank (Brin and Page, 1998).  $P(q|d)$  can be estimated by using a standard language model for IR (Ponte and Croft, 1998).

In summary, Equation (7) is converted to

$$P(e|q) \propto P(\text{type}(t,q))P(p)P(\text{type}(r))P(q|d)P(d). \quad (12)$$

### 3.4 Evidence Merging

We assume that the ranking score of an expert can be acquired by summing up together all scores of the supporting evidences. Thus we calculate experts' scores by aggregating the scores from all evidences as in Equation (1).

## 4 Implementation

The implementation of the proposed model consists of two stages, namely *evidence extraction* and *evidence quality evaluation*.

### 4.1 Evidence Extraction

Recall that we define an evidence for expert search as a quadruple  $\langle \text{topic}, \text{person}, \text{relation}, \text{document} \rangle$ . The evidence extraction covers the extraction of the first three elements, namely *person identification*, *topic discovering* and *relation extraction*.

#### 4.1.1 Person Identification

The occurrences of an expert can be in various forms, such as name and email address. We call each type of form an *expert mask*. Table 1 provides a statistic on various *masks* on the basis of W3C corpus. In Table 1, *rate* is the proportion of the person occurrences with relevant masks to the person occurrences with any of the masks, and *ambiguity* is defined as the probability that a mask is shared by more than one expert.

Mask	Rate/Ambiguity	Sample
Full Name( $N_F$ )	48.2% / 0.0000	Ritu Raj Tiwari
Email Name( $N_E$ )	20.1% / 0.0000	rtiwari@nuance.com
Combined Name ( $N_C$ )	4.2% / 0.3992	Tiwari, Ritu R; R R Tiwari
Abbr. Name( $N_A$ )	21.2% / 0.4890	Ritu Raj ; Ritu
Short Name( $N_S$ )	0.7% / 0.6396	RRT
Alias, new email ( $N_{AE}$ )	7% / 0.4600	Ritiwari rti- wari@hotmail.com

Table 1. Various masks and their ambiguity

1)	Every occurrence of a candidate's <i>email address</i> is normalized to the appropriate <i>candidate id</i> .
2)	Every occurrence of a candidate's <i>full_name</i> is normalized to the appropriate <i>candidate id</i> if there is no ambiguity; otherwise, the occurrence is normalized to the <i>candidate id</i> of the most frequent candidate with that <i>full_name</i> .
3)	Every occurrence of <i>combined name</i> , <i>abbreviated name</i> , and <i>email alias</i> is normalized to the appropriate <i>candidate id</i> if there is no ambiguity; otherwise, the occurrence may be normalized to the <i>candidate id</i> of a candidate whose full name also appears in the document.
4)	All the personal occurrences other than those covered by Heuristic 1) ~ 3) are ignored.

Table 2. Heuristic rules for expert extraction

As Table 1 demonstrates, it is not an easy task to identify all the *masks* with regards to an expert. On one hand, the extraction of *full name* and *email address* is straightforward but suffers from low coverage. On the other hand, the extraction of

*combined name* and *abbreviated name* can complement the coverage, while needs handling of ambiguity.

Table 2 provides the heuristic rules that we use for expert identification. In the step 2) and 3), the rules use *frequency* and *context discourse* for resolving ambiguities respectively. With *frequency*, each expert candidate actually is assigned a prior probability. With *context discourse*, we utilize the intuition that person names appearing similar in a document usually refers to the same person.

#### 4.1.2 Topic Discovering

A queried topic can occur within documents in various forms, too. We use a set of query processing techniques to handle the issue. After the processing, a set of topics transformed from an original query will be obtained and then be used in the search for experts. Table 3 shows five forms of topic discovering from a given query.

Forms	Description	Sample
Phrase Match( $Q_P$ )	The exact match with original query given by users	“ <i>semantic web search engine</i> ”
Bi-gram Match( $Q_B$ )	A set of matches formed by extracting bi-gram of words in the original query	“ <i>semantic web search engine</i> ”
Proximity Match( $Q_{PR}$ )	Each query term appears as a neighborhood within a window of specified size	“ <i>semantic web enhanced search engine</i> ”
Fuzzy Match( $Q_F$ )	A set of matches, each of which resembles the original query in appearance.	“ <i>sementic web seerch engine</i> ”
Stemmed Match( $Q_S$ )	A match formed by stemming the original query.	“ <i>sementic web seerch engin</i> ”

Table 3. Discovered topics from query “semantic web search engine”

#### 4.1.3 Relation Extraction

We focus on extracting relations between topics and expert candidates within a span of a document. To make the extraction easier, we partition a document into a pre-defined layout. Figure 1 provides a template in Backus–Naur form. Figure 2 provides a practical use of the template.

Note that we are not restricting the use of the template only for certain corpus. Actually the template can be applied to many kinds of documents. For example, for web pages, we can construct the <Title> from either the ‘title’ metadata or the con-

tent of web pages (Hu et al., 2006). As for e-mail, we can use the ‘subject’ field as the <Title>.

```

<Document> ::= {<Title>| <Author> |<Body>}
<Title> ::= {<Element>}
<Author> ::= {<Expert>}
<Body> ::= {<Section>}
<Section> ::= {<Section Title><Section Body>}
<SectionTitle> ::= {<Element>}
<SectionBody> ::= {<Section>}|{<Element >}
<Element> ::= <Term>| <Expert>

```

Figure 1. A template of document layout

```

<Title> RDF Primer
<Author> Editors: Frank Manola, fmanola@acm.org
Eric Miller, W3C, em@w3.org
<Body>
  <Section>
    <Section Title>
      2. Making Statements About Resources
    <Section Body>
      RDF is intended to provide a simple way to make state
      These capabilities (the normative specification describe)
      2.1 Basic Concepts
      Imagine trying to state that someone named John Smith
      The form of a simple statement such as:
      ...
  ...

```

Figure 2. An example use of the layout template

With the layout of partitioned documents, we can then explore many types of relations among different blocks. In this paper, we demonstrate the use of five types of relations by extending the study in (Cao et al., 2005).

**Section Relation ( $R_S$ ):** The queried topic and the expert candidate occur in the same <Section>.

**Windowed Section Relation ( $R_{WS}$ ):** The queried topic and the expert candidate occur within a fixed window of a <Section>. In our experiment, we used a window of 200 words.

**Reference Section Relation ( $R_{RS}$ ):** Some <Section>s should be treated specially. For example, the <Section> consisting of reference information like a list of <book, author> can serve as a reliable source connecting a topic and an expert candidate. We call the relation appearing in a special type of <Section> a special reference section relation. It might be argued whether the use of special sections can be generalized. According to our survey, the special <Section>s can be found in various sites such as Wikipedia as well as W3C.

**Title-Author Relation ( $R_{TA}$ ):** The queried topic appears in the <Title> and the expert candidate appears in the <Author>.

**Section Title-Body Relation ( $R_{STB}$ ):** The queried topic and the expert candidate appear in the <Section Title> and <Section Body> of the same <Section>, respectively. Reversely, the queried topic and the expert candidate can appear in the <Section Body> and <Section Title> of a <Section>. The latter case is used to characterize the documents introducing certain expert or the expert introducing certain document.

Note that our model is not restricted to use these five relations. We use them only for the aim of demonstrating the flexibility and effectiveness of fine-grained expert search.

## 4.2 Evidence Quality Evaluation

In this section, we elaborate the mechanism used for evaluating the quality of evidence.

### 4.2.1 Topic-Matching Quality

In Section 4.1.2, we use five techniques in processing query matches, which yield five sets of match types for a given query. Obviously, the different query matches should be associated with different weights because they represent different qualities.

We further note that different bi-grams generated from the same query with the *bi-gram matching* method might also present different qualities. For example, both topic “*css test*” and “*test suite*” are the bi-gram matching for query “*css test suite*”; however, the former might be more informative. To model that, we use the number of returned documents to refine the query weight. The intuition behind that is similar to the thought of IDF popularly used in IR as we prefer to the distinctive bi-grams.

Taking into consideration the above two factors, we calculate the topic-matching quality  $Q_t$  (corresponding to  $P(\text{type}(t,q))$  in Equation (12)) for the given query  $q$  as

$$Q_t = W(\text{type}(t,q)) \frac{\text{MIN}_r(df_r)}{df_t}, \quad (13)$$

where  $t$  means the discovered topic from a document and  $\text{type}(t,q)$  is the matching type between topic  $t$  and query  $q$ .  $W(\text{type}(t,q))$  is the weight for a certain query type,  $df_t$  is the number of returned documents matched by topic  $t$ . In our experiment, we use the 10 training topics of TREC2005 as our training data, and the best quality scores for *phrase match*, *bi-gram match*, *proximity match*, *fuzzy*

*match*, and *stemmed match* are 1, 0.01, 0.05,  $10^{-8}$ , and  $10^{-4}$ , respectively.

### 4.2.2 Person-Matching Quality

An expert candidate can occur in the documents in various ways. The most confident occurrence should be the ones in full name or email address. Others can include last name only, last name plus initial of first name, etc. Thus, the action of rejecting or accepting a person from his/her mask (the surface expression of a person in the text) is not simply a Boolean decision, but a probabilistic one with a reliability weight  $Q_p$  (corresponding to  $P(c|p)$  in Equation (3)). Similarly, the best trained weights for *full name*, *email name*, *combined name*, *abbreviated name*, *short name*, and *alias email* are set to 1, 1, 0.8, 0.2, 0.2, and 0.1, respectively.

### 4.2.3 Relation Type Quality

The relation quality consists of two factors. One factor is about the type of the relation. Different types of relations indicate different strength of the connection between expert candidates and queried topics. In our system, the *section title-body relation* is given the highest confidence. The other factor is about the degree of proximity between a query and an expert candidate. The intuition is that, the more distant are a query and an expert candidate within a relation, the looser the connection between them is. To include these two factors, the quality score  $Q_r$  (corresponding to  $P(\text{type}(r))$  in Equation (12)) of a relation  $r$  is defined as:

$$Q_r = W_r \frac{C_r}{\text{dis}(p,t) + 1}, \quad (14)$$

where  $W_r$  is the weight of relation type  $r$ ,  $\text{dis}(p, t)$  is the distance from the person occurrence  $p$  to the queried topic  $t$  and  $C_r$  is a constant for normalization. Again, we optimize the  $W_r$  based on the training topics, the best weights for *section relation*, *windowed section relation*, *reference section relation*, *title-author relation*, and *section title-body relation* are 1, 4, 10, 45, and 1000 respectively.

### 4.2.4 Document Quality

The quality of evidence also depends on the quality of the document, the context in which it is found. The *document context* can affect the credibility of the evidence in two ways:

**Static quality:** indicating the authority of a document. In our experiment, the static quality  $Q_d$  (corresponding to  $P(d)$  in Equation (12) ) is estimated by the PageRank, which is calculated using a standard iterative algorithm with a damping factor of 0.85 (Brin and Page, 1998).

**Dynamic quality:** by “dynamic”, we mean the quality score varies for different queries  $q$ . We denote the dynamic quality as  $Q_{Dy}(d,q)$  (corresponding to  $P(q|d)$  in Equation (12) ), which is actually the document relevance score returned by a standard language model for IR(Ponte and Croft, 1998).

## 5 Experimental Results

### 5.1 The Evaluation Data

In our experiment, we used the data set in the expert search task of enterprise search track at TREC 2005 and 2006. The document collection is a crawl of the public W3C sites in June 2004. The crawl comprises in total 331,307 web pages. In the following experiments, we used the training set of 10 topics of TREC 2005 for tuning the parameters aforementioned in Section 4.2, and used the test set of 50 topics of TREC 2005 and 49 topics of TREC 2006 as the evaluation data sets.

### 5.2 Evaluation Metrics

We used three measures in evaluation: *Mean average precision (MAP)*, *R-precision (R-P)*, and *Top N precision (P@N)*. They are also the standard measures used in the expert search task of TREC.

### 5.3 Evidence Extraction

In the following experiments, we constructed the baseline by using the query matching methods of *phrase matching*, the expert matching methods of *full name matching and email matching*, and the relation of *section relation*. To show the contribution of each individual method for evidence extraction, we incrementally add the methods to the baseline method. In the following description, we will use ‘+’ to denote applying new method on the previous setting.

#### 5.3.1 Query Matching

Table 4 shows the results of expert search achieved by applying different methods of query matching.

$Q_B$ ,  $Q_{PR}$ ,  $Q_F$ , and  $Q_S$  denote bi-gram match, proximity match, fuzzy match, and stemmed match, respectively. The performance of the proposed model increases stably on MAP when new query matches are added incrementally. We also find that the introduction of  $Q_F$  and  $Q_S$  bring some drop on R-Precision and P@10. It is reasonable because both  $Q_F$  and  $Q_S$  bring high recall while affect the precision a bit. The overall relative improvement of using query matching compared to the baseline is presented in the row “Improv.”. We performed t-tests on MAP. The p-values ( $< 0.05$ ) are presented in the “T-test” row, which shows that the improvement is statistically significant.

	TREC 2005			TREC 2006		
	MAP	R-P	P@10	MAP	R-P	P@10
Baseline	0.1840	0.2136	0.3060	0.3752	0.4585	0.5604
+ $Q_B$	0.1957	0.2438	0.3320	0.4140	0.4910	0.5799
+ $Q_{PR}$	0.2024	<b>0.2501</b>	<b>0.3360</b>	0.4530	<b>0.5137</b>	<b>0.5922</b>
+ $Q_F, Q_S$	<b>0.2030</b>	<b>0.2501</b>	<b>0.3360</b>	<b>0.4580</b>	0.5112	0.5901
Improv.	10.33%	17.09%	9.80%	22.07%	11.49%	5.30%
T-test	0.0084			0.0000		

Table 4. The effects of query matching

#### 5.3.2 Person Matching

For person matching, we considered four types of masks, namely *combined name ( $N_C$ )*, *abbreviated name ( $N_A$ )*, *short name ( $N_S$ )* and *alias and new email ( $N_{AE}$ )*. Table 5 provides the results on person matching at TREC 2005 and 2006. The baseline is the best model achieved in previous section. It seems that there is little improvement on P@10 while an improvement of 6.21% and 14.00% is observed on MAP. This might be due to the fact that the matching method such as  $N_C$  has a higher recall but lower precision.

	TREC 2005			TREC 2006		
	MAP	R-P	P@10	MAP	R-P	P@10
Baseline	0.2030	0.2501	0.3360	0.4580	0.5112	0.5901
+ $N_C$	0.2056	0.2539	<b>0.3463</b>	0.4709	0.5152	0.5931
+ $N_A$	0.2106	0.2545	0.3400	0.5010	0.5181	<b>0.6000</b>
+ $N_S$	0.2111	0.2578	0.3400	0.5121	0.5192	<b>0.6000</b>
+ $N_{AE}$	<b>0.2156</b>	<b>0.2591</b>	0.3400	<b>0.5221</b>	<b>0.5212</b>	<b>0.6000</b>
Improv.	6.21%	3.60%	1.19%	14.00%	1.96%	1.68%
T-test	0.0064			0.0057		

Table 5. The effects of person matching

### 5.3.3 Multiple Relations

For relation extraction, we experimentally demonstrated the use of each of the five relations proposed in Section 4.1.3, i.e., section relation ( $R_S$ ), windowed section relation ( $R_{WS}$ ), reference section relation ( $R_{RS}$ ), title-author relation ( $R_{TA}$ ), and section title-body relation ( $R_{STB}$ ). We used the best model achieved in previous section as the baseline. From Table 6, we can see that the section title-body relation contributes the most to the improvement of the performance. By using all the discovered relations, a significant improvement of 19.94% and 8.35% is achieved.

	TREC 2005			TREC 2006		
	MAP	R-P	P@10	MAP	R-P	P@10
Baseline	0.2156	0.2591	0.3400	0.5221	0.5212	0.6000
+ $R_{WS}$	0.2158	0.2633	0.3380	0.5255	0.5311	0.6082
+ $R_{RS}$	0.2160	0.2630	0.3380	0.5272	0.5314	0.6061
+ $R_{TA}$	0.2234	0.2634	0.3580	0.5354	0.5355	0.6245
+ $R_{STB}$	<b>0.2586</b>	<b>0.3107</b>	<b>0.3740</b>	<b>0.5657</b>	<b>0.5669</b>	<b>0.6510</b>
Improv.	19.94%	19.91%	10.00%	8.35%	8.77%	8.50%
T-test	0.0013			0.0043		

Table 6. The effects of relation extraction

### 5.4 Evidence Quality

The performance of expert search can be further improved by considering the evidence quality. Table 7 shows the results by considering the differences in quality.

We evaluated two kinds of evidence quality: *context static quality* ( $Q_d$ ) and *context dynamic quality* ( $Q_{DY}$ ). Each of the evidence quality contributes about 1%-2% improvement for MAP. The improvement from the PageRank that we calculated from the corpus implies that the web scaled rank technique is also effective in the corpus of documents. Finally, we find a significant relative improvement of 6.13% and 2.86% on MAP by using evidence qualities.

	TREC 2005			TREC 2006		
	MAP	R-P	P@10	MAP	R-P	P@10
Baseline	0.2586	0.3107	0.3740	0.5657	0.5669	0.6510
+ $Q_d$	0.2711	0.3188	0.3720	0.5900	0.5813	0.6796
+ $Q_{DY}$	<b>0.2755</b>	<b>0.3252</b>	<b>0.3880</b>	<b>0.5943</b>	<b>0.5877</b>	<b>0.7061</b>
Improv.	6.13%	4.67%	3.74%	2.86%	3.67%	8.61%
T-test	0.0360			0.0252		

Table 7. The effects of using evidence quality

### 5.5 Comparison with Other Systems

In Table 8, we juxtapose the results of our probabilistic model for fine-grained expert search with automatic expert search systems from the TREC evaluation. The performance of our proposed model is rather encouraging, which achieved comparable results to the best automatic systems on the TREC 2005 and 2006.

		MAP	R-prec	Prec@10
Rank-1 System	TREC2005	0.2749	0.3330	0.4520
	TREC2006 <sup>1</sup>	0.5947	0.5783	0.7041
Our System	TREC2005	0.2755	0.3252	0.3880
	TREC2006	0.5943	0.5877	0.7061

Table 8. Comparison with other systems

## 6 Conclusions

This paper proposed to conduct expert search using a fine-grained level of evidence. Specifically, quadruple evidence was formally defined and served as the basis of the proposed model. Different implementations of evidence extraction and evidence quality evaluation were also comprehensively studied. The main contributions are:

1. The proposal of fine-grained expert search, which we believe to be a promising direction for exploring subtle aspects of evidence.
2. The proposal of probabilistic model for fine-grained expert search. The model facilitates investigating the subtle aspects of evidence.
3. The extensive evaluation of the proposed probabilistic model and its implementation on the TREC data set. The evaluation shows promising expert search results.

In future, we are to explore more domain independent evidences and evaluate the proposed model on the basis of the data from other domains.

## Acknowledgments

The authors would like to thank the three anonymous reviewers for their elaborate and helpful comments. The authors also appreciate the valuable suggestions of Hang Li, Nick Craswell, Yangbo Zhu and Linyun Fu.

<sup>1</sup> This system, where cluster-based re-ranking is used, is a variation of the fine-grained model proposed in this paper.

## References

- Bailey, P., Soboroff, I., Craswell, N., and Vries A.P., Overview of the TREC 2007 Enterprise Track. In: *Proc. of TREC 2007*.
- Balog, K., Azzopardi, L., and Rijke, M. D., 2006. Formal models for expert finding in enterprise corpora. In: *Proc. of SIGIR'06*, pp.43-50.
- Brin, S. and Page, L., 1998. The anatomy of a large-scale hypertextual Web search engine, *Computer Networks and ISDN Systems* (30), pp.107-117.
- Campbell, C.S., Maglio, P., Cozzi, A. and Dom, B., 2003. Expertise identification using email communications. In: *Proc. of CIKM '03* pp.528-531.
- Cao, Y., Liu, J., and Bao, S., and Li, H., 2005. Research on expert search at enterprise track of TREC 2005. In: *Proc. of TREC 2005*.
- Craswell, N., Hawking, D., Vercoustre, A. M. and Wilkins, P., 2001. P@NOPTIC Expert: searching for experts not just for documents. In: *Proc. of Ausweb'01*.
- Craswell, N., Vries, A.P., and Soboroff, I., 2005. Overview of the TREC 2005 Enterprise Track. In: *Proc. of TREC 2005*.
- Davenport, T. H. and Prusak, L., 1998. Working Knowledge: how organizations manage what they know. *Howard Business, School Press*, Boston, MA.
- Dom, B., Eiron, I., Cozzi A. and Yi, Z., 2003. Graph-based ranking algorithms for e-mail expertise analysis, In: *Proc. of SIGMOD'03 workshop on Research issues in data mining and knowledge discovery*.
- Fang, H., Zhou, L., Zhai, C., 2006. Language models for expert finding-UIUC TREC 2006 Enterprise Track Experiments, In: *Proc. of TREC2006*.
- Fu, Y., Xiang, R., Liu, Y., Zhang, M., Ma, S., 2007. A CDD-based Formal Model for Expert Finding. In *Proc. of CIKM 2007*.
- Hertzum, M. and Pejtersen, A. M., 2000. The information-seeking practices of engineers: searching for documents as well as for people. *Information Processing and Management*, 36(5), pp.761-778.
- Hu, Y., Li, H., Cao, Y., Meyerzon, D. Teng, L., and Zheng, Q., 2006. Automatic extraction of titles from general documents using machine learning, *IPM*.
- Kautz, H., Selman, B. and Milewski, A., 1996. Agent amplified communication. In: *Proc. of AAAI'96*, pp. 3-9.
- Mattox, D., Maybury, M. and Morey, D., 1999. Enterprise expert and knowledge discovery. *Technical Report*.
- McDonald, D. W. and Ackerman, M. S., 1998. Just Talk to Me: a field study of expertise location. In: *Proc. of CSCW'98*, pp.315-324.
- Mockus, A. and Herbsleb, J.D., 2002. Expertise Browser: a quantitative approach to identifying expertise, In: *Proc. of ICSE'02*.
- Maconald, C. and Ounis, I., 2006. Voting for candidates: adapting data fusion techniques for an expert search task. In: *Proc. of CIKM'06*, pp.387-396.
- Macdonald, C. and Ounis, I., 2007. Expertise Drift and Query Expansion in Expert Search. In *Proc. of CIKM 2007*.
- Petkova, D., and Croft, W. B., 2006. Hierarchical language models for expert finding in enterprise corpora, In: *Proc. of ICTAI'06*, pp.599-608.
- Ponte, J. and Croft, W., 1998. A language modeling approach to information retrieval, In: *Proc. of SIGIR'98*, pp.275-281.
- Sihn, W. and Heeren F., 2001. Xpertfinder-expert finding within specified subject areas through analysis of e-mail communication. In: *Proc. of the 6th Annual Scientific conference on Web Technology*.
- Soboroff, I., Vries, A.P., and Craswell, N., 2006. Overview of the TREC 2006 Enterprise Track. In: *Proc. of TREC 2006*.
- Steer, L.A. and Lochbaum, K.E., 1988. An expert/expert locating system based on automatic representation of semantic structure, In: *Proc. of the 4th IEEE Conference on Artificial Intelligence Applications*.
- Yimam, D., 1996. Expert finding systems for organizations: domain analysis and the DEMOIR approach. In: *ECSCW'99 workshop of beyond knowledge management: managing expertise*, pp. 276-283.

# Credibility Improves Topical Blog Post Retrieval

**Wouter Weerkamp**

ISLA, University of Amsterdam  
weerkamp@science.uva.nl

**Maarten de Rijke**

ISLA, University of Amsterdam  
mdr@science.uva.nl

## Abstract

Topical blog post retrieval is the task of ranking blog posts with respect to their relevance for a given topic. To improve topical blog post retrieval we incorporate textual credibility indicators in the retrieval process. We consider two groups of indicators: *post level* (determined using information about individual blog posts only) and *blog level* (determined using information from the underlying blogs). We describe how to estimate these indicators and how to integrate them into a retrieval approach based on language models. Experiments on the TREC Blog track test set show that both groups of credibility indicators significantly improve retrieval effectiveness; the best performance is achieved when combining them.

## 1 Introduction

The growing amount of user generated content available online creates new challenges for the information retrieval (IR) community, in terms of search and analysis tasks for this type of content. The introduction of a blog retrieval track at TREC (Ounis et al., 2007) has created a platform where we can begin to address these challenges. During the 2006 edition of the track, two types of blog post retrieval were considered: *topical* (retrieve posts about a topic) and *opinionated* (retrieve opinionated posts about a topic). Here, we consider the former task.

Blogs and blog posts offer unique features that may be exploited for retrieval purposes. E.g., Mishne (2007b) incorporates time in a blog post retrieval model to account for the fact that many blog queries and posts are a response to a news event (Mishne and de Rijke, 2006). Data quality is an issue with blogs—the quality of posts ranges from low to edited news article-like. Some approaches to post retrieval use indirect quality mea-

asures (e.g., elaborate spam filtering (Java et al., 2007) or counting inlinks (Mishne, 2007a)).

Few systems turn the *credibility* (Metzger, 2007) of blog posts into an aspect that can benefit the retrieval process. Our hypothesis is that more credible blog posts are preferred by searchers. The idea of using credibility in the blogosphere is not new: Rubin and Liddy (2006) define a framework for assessing blog credibility, consisting of four main categories: blogger’s expertise and offline identity disclosure; blogger’s trustworthiness and value system; information quality; and appeals and triggers of a personal nature. Under these four categories the authors list a large number of indicators, some of which can be determined from textual sources (e.g., literary appeal), and some of which typically need non-textual evidence (e.g., curiosity trigger); see Section 2.

We give concrete form to Rubin and Liddy (2006)’s indicators and test their impact on blog post retrieval effectiveness. We do not consider all indicators: we only consider indicators that are textual in nature, and to ensure reproducibility of our results, we only consider indicators that can be derived from the TRECBlog06 corpus (and that do not need additional resources such as bloggers’ profiles that may be hard to obtain for technical or legal reasons).

We detail and implement two groups of credibility indicators: *post level* (these use information about individual posts) and *blog level* (these use information from the underlying blogs). Within the post level group, we distinguish between topic dependent and independent indicators. To make matters concrete, consider Figure 1: both posts are relevant to the query “tennis,” but based on obvious surface level features of the posts we quickly determine *Post 2* to be more *credible* than *Post 1*. The most obvious features are spelling errors, the lack of leading capitals, and the large number of exclamation marks and

### Post 1

as for today (monday) we had no school! yaay labor day. but we had tennis from 9-11 at the highschool. after that me suzi melis & ashley had a picnic at cecil park and then played tennis. i just got home right now. it was a very very fun afternoon. (...) we will have a short week. mine will be even shorter b/c i wont be there all day on friday cuz we have the Big 7 Tournament at like keystone oaks or sumthin. so i will miss school the whole day.

### Post 2

Wimbledon champion Venus Williams has pulled out of next week's Kremlin Cup with a knee injury, tournament organisers said on Friday. The American has not played since pulling out injured of last month's China Open. The former world number one has been troubled by various injuries (...) Williams's withdrawal is the latest blow for organisers after Australian Open champion and home favorite Marat Safin withdrew (...).

Figure 1: Two blog posts relevant to the query “tennis.”

personal pronouns—i.e., topic independent ones—and the fact that the language usage in the second post is more easily associated with credible information about tennis than the language usage in the first post—i.e., a topic dependent feature.

Our main finding is that topical blog post retrieval can benefit from using credibility indicators in the retrieval process. Both post and blog level indicator groups each show a significant improvement over the baseline. When we combine all features we obtain the best retrieval performance, and this performance is comparable to the best performing TREC 2006 and 2007 Blog track participants. The improvement over the baseline is stable across most topics, although topic shift occurs in a few cases.

The rest of the paper is organized as follows. In Section 2 we provide information on determining credibility; we also relate previous work to the credibility indicators that we consider. Section 3 specifies our retrieval model, a method for incorporating credibility indicators in our retrieval model, and estimations of credibility indicators. Section 4 gives the results of our experiments aimed at assessing the contribution of credibility towards blog post retrieval effectiveness. We conclude in Section 5.

## 2 Credibility Indicators

In our choice of credibility indicators we use (Rubin and Liddy, 2006)'s work as a reference point. We recall the main points of their framework and relate our indicators to it. We briefly discuss other credibility-related indicators found in the literature.

### 2.1 Rubin and Liddy (2006)'s work

Rubin and Liddy (2006) proposed a four factor analytical framework for blog-readers' credibility assessment of blog sites, based in part on evidentiality theory (Chafe, 1986), website credibility assessment surveys (Stanford et al., 2002), and Van House (2004)'s observations on blog credibility. The four factors—plus indicators for each of them—are:

1. *blogger's expertise and offline identity disclosure* (*a*: name and geographic location; *b*: credentials; *c*: affiliations; *d*: hyperlinks to others; *e*: stated competencies; *f*: mode of knowing);
2. *blogger's trustworthiness and value system* (*a*: biases; *b*: beliefs; *c*: opinions; *d*: honesty; *e*: preferences; *f*: habits; *g*: slogans)
3. *information quality* (*a*: completeness; *b*: accuracy; *c*: appropriateness; *d*: timeliness; *e*: organization (by categories or chronology); *f*: match to prior expectations; *g*: match to information need); and
4. *appeals and triggers of a personal nature* (*a*: aesthetic appeal; *b*: literary appeal (i.e., writing style); *c*: curiosity trigger; *d*: memory trigger; *e*: personal connection).

### 2.2 Our credibility indicators

We only consider credibility indicators that avoid making use of the searcher's or blogger's identity (i.e., excluding 1a, 1c, 1e, 1f, 2e from Rubin and Liddy's list), that can be estimated automatically from available test collections only so as to facilitate repeatability of our experiments (ruling out 3e, 4a, 4c, 4d, 4e), that are textual in nature (ruling out 2d), and that can be reliably estimated with state-of-the-art language technology (ruling out 2a, 2b, 2c, 2g). For reasons that we explain below, we also ignore the “hyperlinks to others” indicator (1d).

The indicators that we do consider—1b, 2f, 3a, 3b, 3c, 3d, 3f, 3g, 4b—are organized in two groups,



depending on the information source that we use to estimate them, *post level* and *blog level*, and the former is further subdivided into *topic independent* and *topic dependent*. Table 1 lists the indicators we consider, together with the corresponding Rubin and Liddy indicator(s).

Let us quickly explain our indicators. First, we consider the use of *capitalization* to be an indicator of good writing style, which in turn contributes to a sense of credibility. Second, we identify Western style *emoticons* (e.g., :-)) and :-D) in blog posts, and assume that excessive use indicates a less credible blog post. Third, words written in all caps are considered *shouting* in a web environment; we consider shouting to be indicative for non-credible posts. Fourth, a credible author should be able to write without (a lot of) *spelling* errors; the more spelling errors occur in a blog post, the less credible we consider it to be. Fifth, we assume that credible texts have a reasonable *length*; the text should supply enough information to convince the reader of the author’s credibility. Sixth, assuming that much of what goes on in the blogosphere is inspired by events in the news (Mishne and de Rijke, 2006), we believe that, for news related topics, a blog post is more credible if it is published around the time of the triggering news event (*timeliness*). Seventh, our *semantic* indicator also exploits the news-related nature of many blog posts, and “prefers” posts whose language usage is similar to news stories on the topic. Eighth, blogs are a popular place for spammers; *spam* blogs are not considered credible and we want to demote them in the search results. Ninth, *comments* are a notable blog feature: readers of a blog post often have the possibility of leaving a comment for other readers or the author. When people comment on a blog post they apparently find the post worth putting effort in, which can be seen as an indicator of credibility (Mishne and Glance, 2006). Tenth, blogs consist of multiple posts in (reverse) chronological order. The temporal aspect of blogs may indicate credibility: we assume that bloggers with an irregular posting behavior are less credible than bloggers who post *regularly*. And, finally, we consider the topical fluctuation of a blogger’s posts. When looking for credible information we would like to retrieve posts from bloggers that have a certain level of (topical) *consistency*: not the fluctuating

indicator	topic dependent?	post level/ blog level	related Rubin & Liddy indicator
capitalization	no	post	4b
emoticons	no	post	4b
shouting	no	post	4b
spelling	no	post	4b
post length	no	post	3a
timeliness	yes	post	3d
semantic	yes	post	3b, 3c
spam	no	blog	3b, 3c, 3f, 3g
comments	no	blog	1b
regularity	no	blog	2f
consistency	no	blog	2f

Table 1: Credibility indicators

behavior of a (personal) blogger, but a solid interest.

### 2.3 Other work

In a web setting, credibility is often couched in terms of authoritativeness and estimated by exploiting the hyperlink structure. Two well-known examples are the PageRank and HITS algorithms (Liu, 2007), that use the link structure in a topic independent or topic dependent way, respectively. Zhou and Croft (2005) propose collection-document distance and signal-to-noise ratio as priors for the indication of quality in web ad hoc retrieval. The idea of using link structure for improving blog post retrieval has been researched, but results do not show improvements. E.g., Mishne (2007a) finds that retrieval performance decreased. This confirms lessons from the TREC web tracks, where participants found no conclusive benefit from the use of link information for ad hoc retrieval tasks (Hawking and Craswell, 2002). Hence, we restrict ourselves to the use of content-based features for blog post retrieval, thus ignoring indicator 1d (hyperlinks to others).

Related to credibility in blogs is the automatic assessment of forum post quality discussed by Weimer et al. (2007). The authors use surface, lexical, syntactic and forum-specific features to classify forum posts as bad posts or good posts. The use of forum-specific features (such as whether or not the post contains HTML, and the fraction of characters that are inside quotes of other posts), gives the highest benefits to the classification. Working in the community question/answering domain, Agichtein et al. (2008) use a content features, as well non-content information available, such as links between items and

explicit quality ratings from members of the community to identify high-quality content.

As we argued above, spam identification may be part of estimating a blog (or blog post’s) credibility. Spam identification has been successfully applied in the blogosphere to improve retrieval effectiveness; see, e.g., (Mishne, 2007b; Java et al., 2007).

### 3 Modeling

In this section we detail the retrieval model that we use, incorporating ranking by relevance and by credibility. We also describe how we estimate the credibility indicators listed in Section 2.

#### 3.1 Baseline retrieval model

We address the baseline retrieval task using a language modeling approach (Croft and Lafferty, 2003), where we rank documents given a query:  $p(d|q) = p(d)p(q|d)p(q)^{-1}$ . Using Bayes’ Theorem we rewrite this, ignoring expressions that do not influence the ranking, obtaining

$$p(d|q) \propto p(d)p(q|d), \quad (1)$$

and, assuming that query terms are independent,

$$p(d|q) \propto p(d) \prod_{t \in q} p(t|\theta_d)^{n(t,q)}, \quad (2)$$

where  $\theta_d$  is the blog post model, and  $n(t, q)$  denotes the number of times term  $t$  occurs in query  $q$ . To prevent numerical underflows, we perform this computation in the log domain:

$$\log p(d|q) \propto \log p(d) + \sum_{t \in q} n(t, q) \log p(t|\theta_d) \quad (3)$$

In our final formula for ranking posts based on relevance only we substitute  $n(t, q)$  by the probability of the term given the query. This allows us to assign different weights to query terms and yields:

$$\log p(d|q) \propto \log p(d) + \sum_{t \in q} p(t|q) \log p(t|\theta_d). \quad (4)$$

For our baseline experiments we assume that all query terms are equally important and set  $p(t|q)$  set to be  $n(t, q) \cdot |q|^{-1}$ . The component  $p(d)$  is the topic independent (“prior”) probability that the document is relevant; in the baseline model, priors are ignored.

#### 3.2 Incorporating credibility

Next, we extend Eq. 4 by incorporating estimations of the credibility indicators listed in Table 1. Recall that our credibility indicators come in two kinds—post level and blog level—and that the post level indicators can be topic independent or topic dependent, while all blog level indicators are topic independent. Now, modeling topic independent indicators is easy—they can simply be incorporated in Eq. 4 as a weighted sum of two priors:

$$p(d) = \lambda \cdot p_{pl}(d) + (1 - \lambda) \cdot p_{bl}(d), \quad (5)$$

where  $p_{pl}(d)$  and  $p_{bl}(d)$  are the post level and blog level prior probability of  $d$ , respectively. The priors  $p_{pl}$  and  $p_{bl}$  are defined as equally weighted sums:

$$\begin{aligned} p_{pl}(d) &= \sum_i \frac{1}{5} \cdot p_i(d) \\ p_{bl}(d) &= \sum_j \frac{1}{4} \cdot p_j(d), \end{aligned}$$

where  $i \in \{\textit{capitalization}, \textit{emoticons}, \textit{shouting}, \textit{spelling}, \textit{post\_length}\}$  and  $j \in \{\textit{spam}, \textit{comments}, \textit{regularity}, \textit{consistency}\}$ . Estimations of the priors  $p_i$  and  $p_j$  are given below; the weighting parameter  $\lambda$  is determined experimentally.

Modeling topic dependent indicators is slightly more involved. Given a query  $q$ , we create a query model  $\theta_q$  that is a mixture of a temporal query model  $\theta_{temporal}$  and a semantic query model  $\theta_{semantic}$ :

$$p(t|\theta_q) = \mu \cdot p(t|\theta_{temporal}) + (1 - \mu) \cdot p(t|\theta_{semantic}). \quad (6)$$

The component models  $\theta_{temporal}$  and  $\theta_{semantic}$  will be estimated below; the parameter  $\mu$  will be estimated experimentally.

Our final ranking formula, then, is obtained by plugging in Eq. 5 and 6 in Eq. 4:

$$\begin{aligned} \log p(d|q) \propto & \log p(d) \\ & + \beta (\sum_t p(t|q) \cdot \log p(t|\theta_d)) \\ & + (1 - \beta) (\sum_t p(t|\theta_q) \cdot \log p(t|\theta_d)). \end{aligned} \quad (7)$$

#### 3.3 Estimating credibility indicators

Next, we specify how each of the credibility indicators is estimated; we do so in two groups: post level and blog level.

### 3.3.1 Post level credibility indicators

**Capitalization** We estimate the capitalization prior as follows:

$$p_{capitalization}(d) = n(c, s) \cdot |s|^{-1}, \quad (8)$$

where  $n(c, s)$  is the number of sentences starting with a capital and  $|s|$  is the number of sentences; we only consider sentences with five or more words.

**Emoticons** The emoticons prior is estimated as

$$p_{emoticons}(d) = 1 - n(e, d) \cdot |d|^{-1}, \quad (9)$$

where  $n(e, d)$  is the number of emoticons in the post and  $|d|$  is the length of the post in words.

**Shouting** We use the following equation to estimate the shouting prior:

$$p_{shouting}(d) = 1 - n(a, d) \cdot |d|^{-1}, \quad (10)$$

where  $n(a, d)$  is the number of all caps words in blog post  $d$  and  $|d|$  is the post length in words.

**Spelling** The spelling prior is estimated as

$$p_{spelling}(d) = 1 - n(m, d) \cdot |d|^{-1}, \quad (11)$$

where  $n(m, d)$  is the number of misspelled (or unknown) words and  $|d|$  is the post length in words.

**Post length** The post length prior is estimated using  $|d|$ , the post length in words:

$$p_{length}(d) = \log(|d|). \quad (12)$$

**Timeliness** We estimate timeliness using the time-based language models  $\theta_{temporal}$  proposed in (Li and Croft, 2003; Mishne, 2007b). I.e., we use a news corpus from the same period as the blog corpus that we use for evaluation purposes (see Section 4.2). We assign a timeliness score per post based on:

$$p(d|\theta_{temporal}) = k^{-1} \cdot (n(date(d), k) + 1), \quad (13)$$

where  $k$  is the number of top results from the initial result list,  $date(d)$  is the date associated with document  $d$ , and  $n(date(d), k)$  is the number of documents in  $k$  with the same date as  $d$ . For our initial result list we perform retrieval on both the blog and the news corpus and take  $k = 50$  for both corpora.

**Semantic** A semantic query model  $\theta_{semantic}$  is obtained using ideas due to Diaz and Metzler (2006). Again, we use a news corpus from the same period as the evaluation blog corpus and estimate  $\theta_{semantic}$ . We issue the query to the external news corpus, retrieve the top 10 documents and extract the top 10 distinctive terms from these documents. These terms are added to the original query terms to capture the language usage around the topic.

### 3.3.2 Blog level credibility indicators

**Spam filtering** To estimate the spaminess of a blog, we take a simple approach. We train an SVM classifier on a labeled splog blog dataset (Kolari et al., 2006) using the top 1500 words for both spam and non-spam blogs as features. For each classified blog  $d$  we have a confidence value  $s(d)$ . If the classifier cannot make a decision ( $s(d) = 0$ ) we set  $p_{spam}(d)$  to 0, otherwise we use the following to transform  $s(d)$  into a spam prior  $p_{spam}(d)$ :

$$p_{spam}(d) = \frac{s(d)}{2|s(d)|} + \frac{-1 \cdot s(d)}{2s(d)^2 + 2|s(d)|} + \frac{1}{2}. \quad (14)$$

**Comments** We estimate the comment prior as

$$p_{comment}(d) = \log(n(r, d)), \quad (15)$$

where  $n(r, d)$  is the number of comments on post  $d$ .

**Regularity** To estimate the regularity prior we use

$$p_{regularity}(d) = \log(\sigma_{interval}), \quad (16)$$

where  $\sigma_{interval}$  expresses the standard deviation of the temporal intervals between two successive posts.

**Topical consistency** Here we use an approach similar to query clarity (Cronen-Townsend and Croft, 2002): based on the list of posts from the same blog we compare the topic distribution of blog  $B$  to the topic distribution in the collection  $C$  and assign a ‘clarity’ value to  $B$ ; a score further away from zero indicates a higher topical consistency. We estimate the topical consistency prior as

$$p_{topic}(d) = \log(clarity(d)), \quad (17)$$

where  $clarity(d)$  is estimated by

$$clarity(d) = \frac{\sum_w p(w|B) \cdot \log\left(\frac{p(w|B)}{p(w)}\right)}{\sum_w p(w|B)} \quad (18)$$

with  $p(w) = \frac{count(w, C)}{|C|}$  and  $p(w|B) = \frac{count(w, B)}{|B|}$ .

### 3.3.3 Efficiency

All estimators discussed above can be implemented efficiently: most are document priors and can therefore be calculated offline. The only topic dependent estimators are *timeliness* and *language usage*; both can be implemented efficiently as specific forms of query expansion.

## 4 Evaluation

In this section we describe the experiments we conducted to answer our research questions about the impact of credibility on blog post retrieval.

### 4.1 Research questions

Our research revolves around the contribution of credibility to the effectiveness of topical blog post retrieval: what is the contribution of individual indicators, of the post level indicators (topic dependent or independent), of the blog level indicators, and of all indicators combined? And do different topics benefit from different indicators? To answer our research question we compared the performance of the baseline retrieval system (as detailed in Section 3.1) with extensions of the baseline system with a single indicator, a set of indicators, or all indicators.

### 4.2 Setup

We apply our models to the TREC Blog06 corpus (Macdonald and Ounis, 2006). This corpus has been constructed by monitoring around 100,000 blog feeds for a period of 11 weeks in early 2006, downloading all posts created in this period. For each permalink (HTML page containing one blog post) the feed id is registered. We can use this id to aggregate post level features to the blog level. In our experiments we use only the HTML documents, 3.2M permalinks, which add up to around 88 GB.

The TREC 2006 and 2007 Blog tracks each offer 50 topics and assessments (Ounis et al., 2007; Macdonald et al., 2007). For topical relevancy, assessment was done using a standard two-level scale: the content of the post was judged to be topically relevant or not. The evaluation metrics that we use are standard ones: mean average precision (MAP) and precision@10 (p@10) (Baeza-Yates and Ribeiro-Neto, 1999). For all our retrieval tasks we use the title field (T) of the topic statement as query.

To estimate the timeliness and semantic credibility indicators, we use AQUAINT-2, a set of newswire articles (2.5 GB, about 907K documents) that are roughly contemporaneous with the TREC Blog06 collection (AQUAINT-2, 2007). Articles are in English and come from a variety of sources.

Statistical significance is tested using a two-tailed paired t-test. Significant improvements over the baseline are marked with  $\Delta$  ( $\alpha = 0.05$ ) or  $\blacktriangle$  ( $\alpha = 0.01$ ). We use  $\nabla$  and  $\blacktriangledown$  for a drop in performance (for  $\alpha = 0.05$  and  $\alpha = 0.01$ , respectively).

### 4.3 Parameter estimation

The models proposed in Section 3.2 contain parameters  $\beta$ ,  $\lambda$  and  $\mu$ . These parameters need to be estimated and, hence, require a training and test set. We use a two-fold parameter estimation process: in the first cycle we estimate the parameters on the TREC 2006 Blog topic set and test these settings on the topics of the TREC 2007 Blog track. The second cycle goes the other way around and trains on the 2007 set, while testing on the 2006 set.

Figure 2 shows the optimum values for  $\lambda$ ,  $\beta$ , and  $\mu$  on the 2006 and the 2007 topic sets for both MAP (bottom lines) and p@10 (top lines). When looking at the MAP scores, the optimal setting for  $\lambda$  is almost identical for the two topic sets: 0.4 for the 2006 set and 0.3 for the 2007 set, and also the optimal setting for  $\beta$  is very similar for both sets: 0.4 for the 2006 set and 0.5 for the 2007 set. As to  $\mu$ , it is clear that timeliness does not improve the performance over using the semantic feature alone and the optimal setting for  $\mu$  is therefore 0.0. Both  $\mu$  and  $\beta$  show similar behavior on p@10 as on MAP, but for  $\lambda$  we see a different trend. If early precision is required, the value of  $\lambda$  should be increased, giving more weight to the topic-independent post level features compared to the blog level features.

### 4.4 Retrieval performance

Table 2 lists the retrieval results for the baseline, for each of the credibility indicators (on top of the baseline), for four subsets of indicators, and for all indicators combined. The baseline performs similar to the median scores at the TREC 2006 Blog track (MAP: 0.2203; p@10: 0.564) and somewhat below the median MAP score at 2007 Blog track (MAP: 0.3340) but above the median p@10 score: 0.3805.

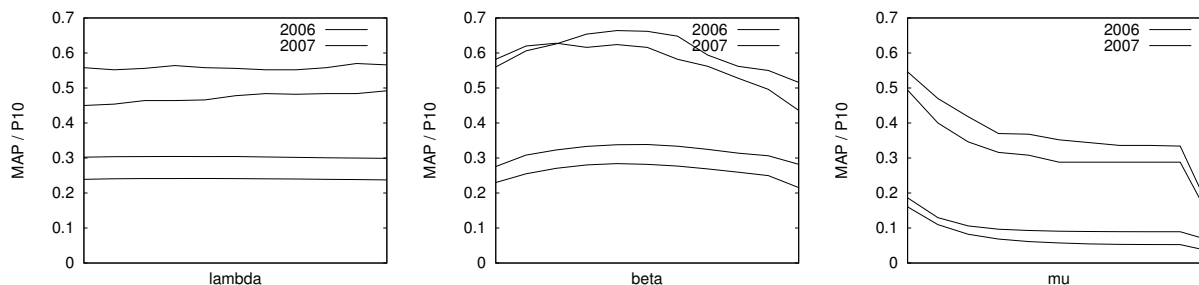


Figure 2: Parameter estimation on the TREC 2006 and 2007 Blog topics. (Left):  $\lambda$ . (Center):  $\beta$ . (Right):  $\mu$ .

	2006		2007	
	map	p@10	map	p@10
baseline	0.2156	0.4360	0.2820	0.5160
capitalization	0.2155	0.4500	0.2824	0.5160
emoticons	0.2156	0.4360	0.2820	0.5200
shouting	0.2159	0.4320	0.2833	0.5100
spelling	0.2179 $\Delta$	0.4480 $\Delta$	0.2839 $\blacktriangle$	0.5220
post length	0.2502 $\blacktriangle$	0.4960 $\blacktriangle$	0.3112 $\blacktriangle$	0.5700 $\blacktriangle$
timeliness	0.1865 $\nabla$	0.4520	0.2660	0.4860
semantic	0.2840 $\blacktriangle$	0.6240 $\blacktriangle$	0.3379 $\blacktriangle$	0.6640 $\blacktriangle$
spam filtering	0.2093	0.4700	0.2814	0.5760 $\blacktriangle$
comments	0.2497 $\blacktriangle$	0.5000 $\blacktriangle$	0.3099 $\blacktriangle$	0.5600 $\blacktriangle$
regularity	0.1658 $\nabla$	0.4940 $\Delta$	0.2353 $\nabla$	0.5640 $\Delta$
consistency	0.2141 $\nabla$	0.4220	0.2785 $\nabla$	0.5040
post level (topic indep.)	0.2374 $\blacktriangle$	0.4920 $\blacktriangle$	0.2990 $\blacktriangle$	0.5660 $\blacktriangle$
post level (topic dep.)	0.2840 $\blacktriangle$	0.6240 $\blacktriangle$	0.3379 $\blacktriangle$	0.6640 $\blacktriangle$
post level (all)	0.2911 $\blacktriangle$	0.6380 $\blacktriangle$	0.3369 $\blacktriangle$	0.6620 $\blacktriangle$
blog level	0.2391 $\blacktriangle$	0.4500	0.3023 $\blacktriangle$	0.5580 $\blacktriangle$
all	0.3051 $\blacktriangle$	0.6880 $\blacktriangle$	0.3530 $\blacktriangle$	0.6900 $\blacktriangle$

Table 2: Retrieval performance on 2006 and 2007 topics, using  $\lambda = 0.3$ ,  $\beta = 0.4$ , and  $\mu = 0.0$ .

Some (topic independent) post level indicators hurt the MAP score, while others help (for both years, and both measures). Combined, the topic independent post level indicators perform less well than the use of one of them (*post length*). As to the topic dependent post level indicators, *timeliness* hurts performance on MAP for both years, while the semantic indicator provides significant improvements across the board (resulting in a top 2 score in terms of MAP and a top 5 score in terms of p@10, when compared to the TREC 2006 Blog track participants that only used the T field).

Some of the blog level features hurt more than they help (*regularity*, *consistency*), while the *comments* feature helps, on all measures, and for both years. Combined, the blog level features help less

than the use of one of them (*comments*).

As a group, the combined post level features help more than either of the two post level sub groups alone. The blog level features show similar results to the topic-independent post level features, obtaining a significant increase on both MAP and p@10, but lower than the topic-dependent post level features.

The grand combination of all credibility indicators leads to a significant improvement over any of the single indicators and over any of the four subsets considered in Table 2. The MAP score of this run is higher than the best performing run in the TREC 2006 Blog track and has a top 3 performance on p@10; its 2007 performance is just within the top half on both MAP and p@10.

#### 4.5 Analysis

Next we examine the differences in average precision (per topic) between the baseline and subsets of indicators (post and blog level) and the grand combination. We limit ourselves to an analysis of the MAP scores. Figure 3 displays the per topic average precision scores, where topics are sorted by absolute gain of the grand combination over the baseline.

In 2006, 7 (out of 50) topics were negatively affected by the use of credibility indicators; in 2007, 15 (out of 50) were negatively affected. Table 3 lists the topics that displayed extreme behavior (in terms of relative performance gain or drop in AP score). While the extreme drops for both years are in the same range, the gains for 2006 are more extreme than for 2007.

The topic that is hurt most (in absolute terms) by the credibility indicators is the 2007 topic 910: *aperto network* (AP -0.2781). The semantic indicator is to blame for this decrease is: the terms included in the expanded query shift the topic from a wireless broadband provider to television networks.

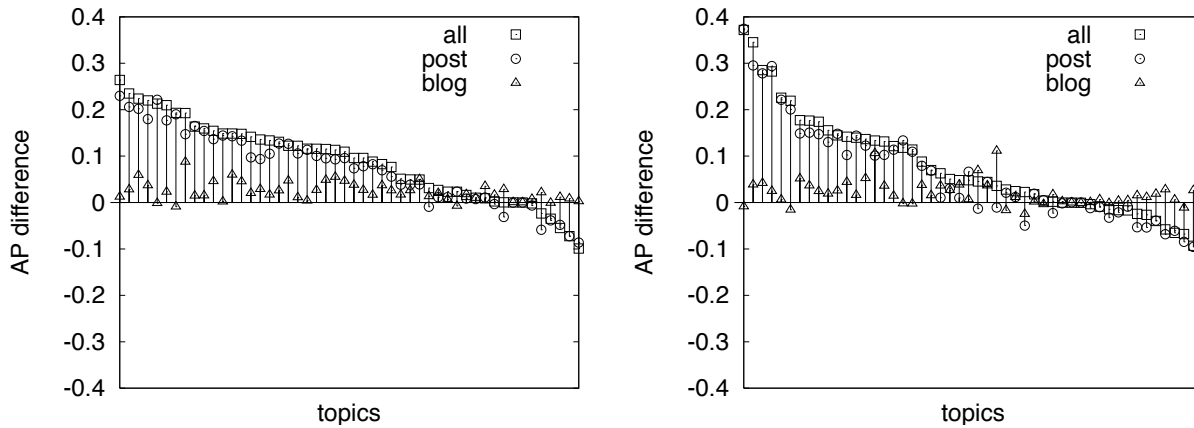


Figure 3: Per-topic AP differences between baseline run and runs with blog level features (triangles), post level features (circles) and all feature (squares) on the 2006 (left) en 2007 (right) topics.

Table 3: Extreme performance gains/drops of the grand combination over the baseline (MAP).

2006		
id	topic	% gain/loss
900	mcdonalds	+525.9%
866	foods	+446.2%
865	basque	+308.6%
862	blackberry	-21.5%
870	barry bonds	-35.2%
898	business intelligence resources	-78.8%
2007		
id	topic	% gain/loss
923	challenger	+162.1%
926	hawthorne heights	+160.7%
945	bolivia	+125.5%
943	censure	-49.4%
928	big love	-80.0%
904	alterman	-84.2%

Topics that gain most (in absolute terms) are 947 (*sasha cohen*; AP +0.3809) and 923 (*challenger*; AP +0.3622) from the 2007 topic set.

Finally, the combination of all credibility indicators hurts 7 (2006) plus 15 (2007) equals 22 topics; for the post level indicators get a performance drop in AP for 28 topics (10 plus 18, respectively) and for the blog level indicators we get a drop for 15 topics (4 plus 11, respectively). Hence, the combination of all indicators strikes a good balance between overall performance gain and per topic risk.

## 5 Conclusions

We provided efficient estimations for 11 credibility indicators and assessed their impact on topical blog post retrieval, on top of a content-based retrieval

baseline. We compared the contribution of these indicators, both individually and in groups, and found that (combined) they have a significant positive impact on topical blog post retrieval effectiveness. Certain single indicators, like *post length* and *comments*, make good credibility indicators on their own; the best performing credibility indicator group consists of topic dependent post level ones. Other future work concerns indicator selection: instead of taking all indicators on board, consider selected indicators only, in a topic dependent fashion.

Our choice of credibility indicators was based on a framework proposed by Rubin and Liddy (2006): the estimators we used are natural implementations of the selected indicators, but by no means the only possible ones. In future work we intend to extend the set of indicators considered so as to include, e.g., stated competencies (1e), by harvesting and analyzing bloggers' profiles, and to extend the set of estimators for indicators that we already consider such as reading level measures (e.g., Flesch-Kincaid) for the literary appeal indicator (4b).

## Acknowledgments

We would like to thank our reviewers for their feedback. Both authors were supported by the E.U. IST programme of the 6th FP for RTD under project MultiMATCH contract IST-033104. De Rijke was also supported by NWO under project numbers 017.001.190, 220-80-001, 264-70-050, 354-20-005, 600.065.120, 612-13-001, 612.000.106, 612.066.-302, 612.069.006, 640.001.501, and 640.002.501.

## References

- Agichtein, E., Castillo, C., Donato, D., Gionis, A., and Mishne, G. (2008). Finding high-quality content in social media. In *WSDM '08*.
- AQUAINT-2 (2007). URL: [http://trec.nist.gov/data/qa/2007\\_qadata/qa.07.guidelines.html#documents](http://trec.nist.gov/data/qa/2007_qadata/qa.07.guidelines.html#documents).
- Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.
- Chafe, W. (1986). Evidentiality in English conversation and academic writing. In Chaf, W. and Nichols, J., editors, *Evidentiality: The Linguistic Coding of Epistemology*, volume 20, pages 261–273. Ablex Publishing Corporation.
- Croft, W. B. and Lafferty, J., editors (2003). *Language Modeling for Information Retrieval*. Kluwer.
- Cronen-Townsend, S. and Croft, W. (2002). Quantifying query ambiguity. In *Proceedings of Human Language Technology 2002*, pages 94–98.
- Diaz, F. and Metzler, D. (2006). Improving the estimation of relevance models using large external corpora. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, New York. ACM Press.
- Hawking, D. and Craswell, N. (2002). Overview of the TREC-2001 web track. In *The Tenth Text Retrieval Conferences (TREC-2001)*, pages 25–31.
- Java, A., Kolari, P., Finin, T., Joshi, A., and Martineau, J. (2007). The blogvox opinion retrieval system. In *The Fifteenth Text REtrieval Conference (TREC 2006)*.
- Kolari, P., Finin, T., Java, A., and Joshi, A. (2006). Splog blog dataset. URL: <http://ebiquity.umbc.edu/resource/html/id/212/Splog-Blog-Dataset>.
- Li, X. and Croft, W. (2003). Time-based language models. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM)*, pages 469–475.
- Liu, B. (2007). *Web Data Mining*. Springer-Verlag, Heidelberg.
- Macdonald, C. and Ounis, I. (2006). The trec blogs06 collection: Creating and analyzing a blog test collection. Technical Report TR-2006-224, Department of Computer Science, University of Glasgow.
- Macdonald, C., Ounis, I., and Soboroff, I. (2007). Overview of the trec 2007 blog track. In *TREC 2007 Working Notes*, pages 31–43.
- Metzger, M. (2007). Making sense of credibility on the web: Models for evaluating online information and recommendations for future research. *Journal of the American Society for Information Science and Technology*, 58(13):2078–2091.
- Mishne, G. (2007a). *Applied Text Analytics for Blogs*. PhD thesis, University of Amsterdam, Amsterdam.
- Mishne, G. (2007b). Using blog properties to improve retrieval. In *Proceedings of ICWSM 2007*.
- Mishne, G. and de Rijke, M. (2006). A study of blog search. In Lalmas, M., MacFarlane, A., R ger, S., Tombros, A., Tsirikla, T., and Yavlinsky, A., editors, *Advances in Information Retrieval: Proceedings 28th European Conference on IR Research (ECIR 2006)*, volume 3936 of LNCS, pages 289–301. Springer.
- Mishne, G. and Glance, N. (2006). Leave a reply: An analysis of weblog comments. In *Proceedings of WWW 2006*.
- Ounis, I., de Rijke, M., Macdonald, C., Mishne, G., and Soboroff, I. (2007). Overview of the trec-2006 blog track. In *The Fifteenth Text REtrieval Conference (TREC 2006) Proceedings*.
- Rubin, V. and Liddy, E. (2006). Assessing credibility of weblogs. In *Proceedings of the AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs (CAAW)*.
- Stanford, J., Tauber, E., Fogg, B., and Marable, L. (2002). Experts vs online consumers: A comparative credibility study of health and finance web sites. URL: [http://www.consumerwebwatch.org/news/report3\\_credibilityresearch/slicedbread.pdf](http://www.consumerwebwatch.org/news/report3_credibilityresearch/slicedbread.pdf).
- Van House, N. (2004). Weblogs: Credibility and collaboration in an online world. URL: [people.ischool.berkeley.edu/~vanhouse/Van%20House%20trust%20workshop.pdf](http://people.ischool.berkeley.edu/~vanhouse/Van%20House%20trust%20workshop.pdf).
- Weimer, M., Gurevych, I., and Mehlhauser, M. (2007). Automatically assessing the post quality in online discussions on software. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 125–128.
- Zhou, Y. and Croft, W. B. (2005). Document quality models for web ad hoc retrieval. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 331–332.

# Linguistically Motivated Features for Enhanced Back-of-the-Book Indexing

Andras Csomai and Rada Mihalcea

Department of Computer Science

University of North Texas

csomaia@unt.edu, rada@cs.unt.edu

## Abstract

In this paper we present a supervised method for back-of-the-book index construction. We introduce a novel set of features that goes beyond the typical frequency-based analysis, including features based on discourse comprehension, syntactic patterns, and information drawn from an online encyclopedia. In experiments carried out on a book collection, the method was found to lead to an improvement of roughly 140% as compared to an existing state-of-the-art supervised method.

## 1 Introduction

Books represent one of the oldest forms of written communication and have been used since thousands of years ago as a means to store and transmit information. Despite this fact, given that a large fraction of the electronic documents available online and elsewhere consist of short texts such as Web pages, news articles, scientific reports, and others, the focus of natural language processing techniques to date has been on the automation of methods targeting short documents. We are witnessing however a change: more and more books are becoming available in electronic format, in projects such as the Million Books project (<http://www.archive.org/details/millionbooks>), the Gutenberg project (<http://www.gutenberg.org>), or Google Book Search (<http://books.google.com>). Similarly, a large number of the books published in recent years are often available – for purchase or through libraries – in electronic format. This means that the need for language processing techniques able to handle very large documents such as books is becoming increasingly important.

This paper addresses the problem of automatic back-of-the-book index construction. A back-of-the-book index typically consists of the most important keywords addressed in a book, with pointers to the relevant pages inside the book. The construction of such indexes is one of the few tasks related to publishing that still requires extensive human labor. Although there is a certain degree of computer assistance, consisting of tools that help the professional indexer to organize and edit the index, there are no methods that would allow for a complete or nearly-complete automation.

In addition to helping professional indexers in their task, an automatically generated back-of-the-book index can also be useful for the automatic storage and retrieval of a document; as a quick reference to the content of a book for potential readers, researchers, or students (Schutze, 1998); or as a starting point for generating ontologies tailored to the content of the book (Feng et al., 2006).

In this paper, we introduce a supervised method for back-of-the-book index construction, using a novel set of linguistically motivated features. The algorithm learns to automatically identify important keywords in a book based on an ensemble of syntactic, discourse-based and information-theoretic properties of the candidate concepts. In experiments performed on a collection of books and their indexes, the method was found to exceed by a large margin the performance of a previously proposed state-of-the-art supervised system for keyword extraction.

## 2 Supervised Back-of-the-Book Indexing

We formulate the problem of back-of-the-book indexing as a supervised keyword extraction task, by making a binary yes/no classification decision at the



level of each candidate index entry. Starting with a set of candidate entries, the algorithm automatically decides which entries should be added to the back-of-the-book index, based on a set of linguistic and information theoretic features. We begin by identifying the set of candidate index entries, followed by the construction of a feature vector for each such candidate entry. In the training data set, these feature vectors are also assigned with a correct label, based on the presence/absence of the entry in the gold standard back-of-the-book index provided with the data. Finally, a machine learning algorithm is applied, which automatically classifies the candidate entries in the test data for their likelihood to belong to the back-of-the-book index.

The application of a supervised algorithm requires three components: a data set, which is described next; a set of features, which are described in Section 3; and a machine learning algorithm, which is presented in Section 4.

## 2.1 Data

We use a collection of books and monographs from the eScholarship Editions collection of the University of California Press (*UC Press*),<sup>1</sup> consisting of 289 books, each with a manually constructed back-of-the-book index. The average length of the books in this collection is 86053 words, and the average length of the indexes is 820 entries. A collection of 56 books was previously introduced in (Csomai and Mihalcea, 2006); however, that collection is too small to be split in training and test data to support supervised keyword extraction experiments.

The UC Press collection was provided in a standardized XML format, following the Text Encoding Initiative (TEI) recommendations, and thus it was relatively easy to process the collection and separate the index from the body of the text.

In order to use this corpus as a gold standard collection for automatic index construction, we had to eliminate the *inversions*, which are typical in human-built indexes. *Inversion* is a method used by professional indexers by which they break the ordering of the words in each index entry, and list the head first, thereby making it easier to find entries in an alphabetically ordered index. As an example, consider the entry *indexing of illustrations*, which, following inversion, becomes *illustrations, indexing of*. To eliminate inversion, we use an approach that gen-

erates each permutation of the composing words for each index entry, looks up the frequency of that permutation in the book, and then chooses the one with the highest frequency as the correct reconstruction of the entry. In this way, we identify the form of the index entries as appearing in the book, which is the form required for the evaluation of extraction methods. Entries that cannot be found in the book, which were most likely generated by the human indexers, are preserved in their original ordering.

For training and evaluation purposes, we used a random split of the collection into 90% training and 10% test. This yields a training corpus of 259 documents and a test data set of 30 documents.

## 2.2 Candidate Index Entries

Every sequence of words in a book represents a potential candidate for an entry in the back-of-the-book index. Thus, we extract from the training and the test data sets all the n-grams (up to the length of four), not crossing sentence boundaries. These represent the candidate index entries that will be used in the classification algorithm. The training candidate entries are then labeled as positive or negative, depending on whether the given n-gram was found in the back-of-the-book index associated with the book.

Using a n-gram-based method to extract candidate entries has the advantage of providing high coverage, but the unwanted effect of producing an extremely large number of entries. In fact, the resulting set is unmanageably large for any machine learning algorithm. Moreover, the set is extremely unbalanced, with a ratio of positive and negative examples of 1:675, which makes it unsuitable for most machine learning algorithms. In order to address this problem, we had to find ways to reduce the size of the data set, possibly eliminating the training instances that will have the least negative effect on the usability of the data set.

The first step to reduce the size of the data set was to use the candidate filtering techniques for unsupervised back-of-the-book index construction that we proposed in (Csomai and Mihalcea, 2007). Namely, we use the commonword and comma filters, which are applied to both the training and the test collections. These filters work by eliminating all the n-grams that begin or end with a common word (we use a list of 300 most frequent English words), as well as those n-grams that cross a comma. This results in a significant reduction in the number of neg-

<sup>1</sup><http://content.cdlib.org/escholarship/>

	positive	negative	total	positive:negative ratio
Training data				
All (original)	71,853	48,499,870	48,571,723	1:674.98
Commonword/comma filters	66,349	11,496,661	11,563,010	1:173.27
10% undersampling	66,349	1,148,532	1,214,881	1:17.31
Test data				
All (original)	7,764	6,157,034	6,164,798	1:793.02
Commonword/comma filters	7,225	1,472,820	1,480,045	1:203.85

Table 1: Number of training and test instances generated from the UC Press data set

ative examples, from 48 to 11 million instances, with a loss in terms of positive examples of only 7.6%.

The second step is to use a technique for balancing the distribution of the positive and the negative examples in the data sets. There are several methods proposed in the existing literature, focusing on two main solutions: undersampling and oversampling (Weiss and Provost, 2001). Undersampling (Kubat and Matwin, 1997) means the elimination of instances from the majority class (in our case negative examples), while oversampling focuses on increasing the number of instances of the minority class. Aside from the fact that oversampling has hard to predict effects on classifier performance, it also has the additional drawback of increasing the size of the data set, which in our case is undesirable. We thus adopted an undersampling solution, where we randomly select 10% of the negative examples. Evidently, the undersampling is applied only to the training set.

Table 1 shows the number of positive and negative entries in the data set, for the different pre-processing and balancing phases.

### 3 Features

An important step in the development of a supervised system is the choice of features used in the learning process. Ideally, any property of a word or a phrase indicating that it could be a good keyword should be represented as a feature and included in the training and test examples. We use a number of features, including information-theoretic features previously used in unsupervised keyword extraction, as well as a novel set of features based on syntactic and discourse properties of the text, or on information extracted from external knowledge repositories.

#### 3.1 Phraseness and Informativeness

We use the phraseness and informativeness features that we previously proposed in (Csomai and Mihalcea, 2007). Phraseness refers to the degree to which

a sequence of words can be considered a phrase. We use it as a measure of lexical cohesion of the component terms and treat it as a collocation discovery problem. Informativeness represents the degree to which the keyphrase is representative for the document at hand, and it correlates to the amount of information conveyed to the user.

To measure the **informativeness** of a keyphrase, various methods can be used, some of which were previously proposed in the keyword extraction literature:

- *tf.idf*, which is the traditional information retrieval metric (Salton and Buckley, 1997), employed in most existing keyword extraction applications. We measure inverse document frequency using the article collection of the online encyclopedia Wikipedia.
- $\chi^2$  *independence test*, which measures the degree to which two events happen together more often than by chance. In our work, we use the  $\chi^2$  in a novel way. We measure the informativeness of a keyphrase by finding if a phrase occurs in the document more frequently than it would by chance. The information required for the  $\chi^2$  independence test can be typically summed up in a contingency table (Manning and Schutze, 1999):

count(phrase in document)	count(all other phrases in document)
count(phrase in other documents)	count(all other phrases in all other documents)

The independence score is calculated based on the observed ( $O$ ) and expected ( $E$ ) counts:

$$\chi^2 = \sum_{i,j} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

where  $i, j$  are the row and column indices of the

contingency table. The  $O$  counts are the cells of the table. The  $E$  counts are calculated from the marginal probabilities (the sum of the values of a column or a row) converted into proportions by dividing them with the total number of observed events ( $N$ ):

$$N = O_{1,1} + O_{1,2} + O_{2,1} + O_{2,2}$$

Then the expected count for seeing the phrase in the document is:

$$E_{1,1} = \frac{O_{1,1} + O_{1,2}}{N} \times \frac{O_{1,1} + O_{2,1}}{N} \times N$$

To measure the **phraseness** of a candidate phrase we use a technique based on the  $\chi^2$  independence test. We measure the independence of the events of seeing the components of the phrase in the text. This method was found to be one of the best performing models in collocation discovery (Pecina and Schlesinger, 2006). For n-grams where  $N > 2$  we apply the  $\chi^2$  independence test by splitting the phrase in two (e.g. for a 4-gram, we measure the independence of the composing bigrams).

### 3.2 Discourse Comprehension Features

Very few existing keyword extraction methods look beyond word frequency. Except for (Turney and Littman, 2003), who uses pointwise mutual information to improve the coherence of the keyword set, we are not aware of any other work that attempts to use the semantics of the text to extract keywords. The fact that most systems rely heavily on term frequency properties poses serious difficulties, since many index entries appear only once in the document, and thus cannot be identified by features based solely on word counts. For instance, as many as 52% of the index entries in our training data set appeared only once in the books they belong to. Moreover, another aspect not typically covered by current keyword extraction methods is the coherence of the keyword set, which can also be addressed by discourse-based properties.

In this section, we propose a novel feature for keyword extraction inspired by work on discourse comprehension. We use a construction integration framework, which is the backbone used by many discourse comprehension theories.

#### 3.2.1 Discourse Comprehension

Discourse comprehension is a field in cognitive science focusing on the modeling of mental pro-

cesses associated with reading and understanding text. The most widely accepted theory for discourse comprehension is the construction integration theory (Kintsch, 1998). According to this theory, the elementary units of comprehension are propositions, which are defined as instances of a predicate-argument schema. As an example, consider the sentence *The hemoglobin carries oxygen*, which generates the predicate CARRY[HEMOGLOBIN,OXIGEN]. The processing cycle of the construction integration model processes one proposition at a time, and builds a local representation of the text in the working memory, called the propositional network.

During the *construction* phase, propositions are extracted from a segment of the input text (typically a single sentence) using linguistic features. The propositional network is represented as a graph, with nodes consisting of propositions, and weighted edges representing the semantic relations between them. All the propositions generated from the input text are inserted into the graph, as well as all the propositions stored in the short term memory. The short term memory contains the propositions that compose the representation of the previous few sentences. The second phase of the construction step is the addition of past experiences (or background knowledge), which is stored in the long term memory. This is accomplished by adding new elements to the graph, usually consisting of the set of closely related propositions from the long term memory.

After processing a sentence, the *integration* step establishes the role of each proposition in the meaning representation of the current sentence, through a spreading activation applied on the propositions derived from the original sentence. Once the weights are stabilized, the set of propositions with the highest activation values give the mental representation of the processed sentence. The propositions with the highest activation values are added to the short term memory, the working memory is cleared and the process moves to the next sentence. Figure 3.2.1 shows the memory types used in the construction integration process and the main stages of the process.

#### 3.2.2 Keyword Extraction using Discourse Comprehension

The main purpose of the short term memory is to ensure the coherence of the meaning representation across sentences. By keeping the most important propositions in the short term memory, the spreading activation process transfers additional weight to se-

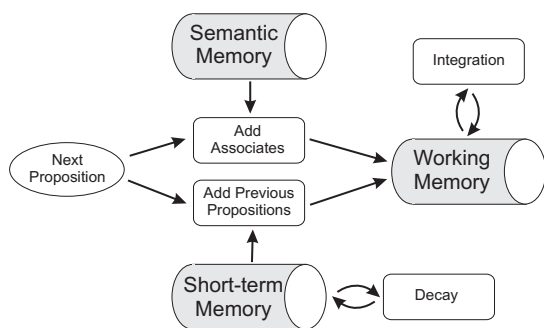


Figure 1: The construction integration process

manically related propositions in the sentences that follow. This also represents a way of alleviating one of the main problems of statistical keyword extraction, namely the sole dependence on term frequency. Even if a phrase appears only once, the construction integration process ensures the presence of the phrase in the short term memory as long as it is relevant to the current topic, thus being a good indicator of the phrase importance.

The construction integration model is not directly applicable to keyword extraction due to a number of practical difficulties. The first implementation problem was the lack of a propositional parser. We solve this problem by using a shallow parser to extract noun phrase chunks from the original text (Munoz et al., 1999). Second, since spreading activation is a process difficult to control, with several parameters that require fine tuning, we use instead a different graph centrality measure, namely PageRank (Brin and Page, 1998).

Finally, to represent the relations inside the long term semantic memory, we use a variant of latent semantic analysis (LSA) (Landauer et al., 1998) as implemented in the InfoMap package,<sup>2</sup> trained on a corpus consisting of the British National Corpus, the English Wikipedia, and the books in our collection. To alleviate the data sparsity problem, we also use the pointwise mutual information (PMI) to calculate the relatedness of the phrases based on the book being processed.

The final system works by iterating the following steps: (1) Read the text sentence by sentence. For each new sentence, a graph is constructed, consisting of the noun phrase chunks extracted from the original text. This set of nodes is augmented with all the phrases from the short term memory. (2) A

weighted edge is added between all the nodes, based on the semantic relatedness measured between the phrases by using LSA and PMI. We use a weighted combination of these two measures, with a weight of 0.9 assigned to LSA and 0.1 to PMI. For the nodes from the short term memory, we adjust the connection weights to account for memory decay, which is a function of the distance from the last occurrence. We implement decay by decreasing the weight of both the outgoing and the incoming edges by  $n * \alpha$ , where  $n$  is the number of sentences since we last saw the phrase and  $\alpha = 0.1$ . (3) Apply PageRank on the resulting graph. (4) Select the 10 highest ranked phrases and place them in the short term memory. (5) Read the next sentence and go back to step (1).

Three different features are derived based on the construction integration model:

- **CI short term memory frequency** (*CI short-term*), which measures the number of iterations that the phrase remains in the short term memory, which is seen as an indication of the phrase importance.
- **CI normalized short term memory frequency** (*CI normalized*), which is the same as *CI short-term*, except that it is normalized by the frequency of the phrase. Through this normalization, we hope to enhance the effect of the semantic relatedness of the phrase to subsequent sentences.
- **CI maximum score** (*CI maxscore*), which measures the maximum centrality score the phrase achieves across the entire book. This can be thought of as a measure of the importance of the phrase in a smaller coherent segment of the document.

### 3.3 Syntactic Features

Previous work has pointed out the importance of syntactic features for supervised keyword extraction (Hulth, 2003). The construction integration model described before is already making use of syntactic patterns to some extent, through the use of a shallow parser to identify noun phrases. However, that approach does not cover patterns other than noun phrases. To address this limitation, we introduce a new feature that captures the part-of-speech of the words composing a candidate phrase.

<sup>2</sup><http://infomap.stanford.edu/>

There are multiple ways to represent such a feature. The simplest is to create a string feature consisting of the concatenation of the part-of-speech tags. However, this representation imposes limitations on the machine learning algorithms that can be used, since many learning systems cannot handle string features. The second solution is to introduce a binary feature for each part-of-speech tag pattern found in the training and the test data sets. In our case this is again unacceptable, given the size of the documents we work with and the large number of syntactic patterns that can be extracted. Instead, we decided on a novel solution which, rather than using the part-of-speech pattern directly, determines the probability of a phrase with a certain tag pattern to be selected as a keyphrase. Formally:

$$P(\text{pattern}) = \frac{C(\text{pattern}, \text{positive})}{C(\text{pattern})}$$

where  $C(\text{pattern}, \text{positive})$  is the number of distinct phrases having the tag pattern  $\text{pattern}$  and being selected as keyword, and  $C(\text{pattern})$  represents the number of distinct phrases having the tag pattern  $\text{pattern}$ . This probability is estimated based on the training collection, and is used as a numeric feature. We refer to this feature as *part-of-speech pattern*.

### 3.4 Encyclopedic Features

Recent work has suggested the use of domain knowledge to improve the accuracy of keyword extraction. This is typically done by consulting a vocabulary of plausible keyphrases, usually in the form of a list of subject headings or a domain specific thesaurus. The use of a vocabulary has the additional benefit of eliminating the extraction of incomplete phrases (e.g. "States of America"). In fact, (Medelyan and Witten, 2006) reported an 110% F-measure improvement in keyword extraction when using a domain-specific thesaurus.

In our case, since the books can cover several domains, the construction and use of domain-specific thesauruses is not plausible, as the advantage of such resources is offset by the time it usually takes to build them. Instead, we decided to use encyclopedic information, as a way to ensure high coverage in terms of domains and concepts.

We use Wikipedia, which is the largest and the fastest growing encyclopedia available today, and whose structure has the additional benefit of being particularly useful for the task of keyword extrac-

tion. Wikipedia includes a rich set of links that connect important phrases in an article to their corresponding articles. These links are added manually by the Wikipedia contributors, and follow the general guidelines of annotation provided by Wikipedia. The guidelines coincide with the goals of keyword extraction, and thus the Wikipedia articles and their link annotations can be treated as a vast keyword annotated corpus.

We make use of the Wikipedia annotations in two ways. First, if a phrase is used as the title of a Wikipedia article, or as the anchor text in a link, this is a good indicator that the given phrase is well formed. Second, we can also estimate the probability of a term  $W$  to be selected as a keyword in a new document by counting the number of documents where the term was already selected as a keyword ( $\text{count}(D_{\text{key}})$ ) divided by the total number of documents where the term appeared ( $\text{count}(D_W)$ ). These counts are collected from the entire set of Wikipedia articles.

$$P(\text{keyword}|W) \approx \frac{\text{count}(D_{\text{key}})}{\text{count}(D_W)} \quad (1)$$

This probability can be interpreted as "the more often a term was selected as a keyword among its total number of occurrences, the more likely it is that it will be selected again." In the following, we will refer to this feature as *Wikipedia keyphraseness*.

### 3.5 Other Features

In addition to the features described before, we add several other features frequently used in keyword extraction: the frequency of the phrase inside the book (*term frequency (tf)*); the number of documents that include the phrase (*document frequency (df)*); a combination of the two (*tf.idf*); the within-document frequency, which divides a book into ten equally-sized segments, and counts the number of segments that include the phrase (*within document frequency*); the length of the phrase (*length of phrase*); and finally a binary feature indicating whether the given phrase is a named entity, according to a simple heuristic based on word capitalization.

## 4 Experiments and Evaluation

We integrate the features described in the previous section in a machine learning framework. The system is evaluated on the data set described in Section 2.1, consisting of 289 books, randomly split into

90% training (259 books) and 10% test (30 books). We experiment with three learning algorithms, selected for the diversity of their learning strategy: multilayer perceptron, SVM, and decision trees. For all three algorithms, we use their implementation as available in the Weka package.

For evaluation, we use the standard information retrieval metrics: precision, recall, and F-measure. We use two different mechanisms for selecting the number of entries in the index. In the first evaluation (*ratio-based*), we use a fixed ratio of 0.45% from the number of words in the text; for instance, if a book has 100,000 words, the index will consist of 450 entries. This number was estimated based on previous observations regarding the typical size of a back-of-the-book index (Csomai and Mihalcea, 2006). In order to match the required number of entries, we sort all the candidates in reversed order of the confidence score assigned by the machine learning algorithm, and consequently select the top entries in this ranking. In the second evaluation (*decision-based*), we allow the machine learning algorithm to decide on the number of keywords to extract. Thus, in this evaluation, all the candidates labeled as keywords by the learning algorithm will be added to the index. Note that all the evaluations are run using a training data set with 10% undersampling of the negative examples, as described before.

Table 2 shows the results of the evaluation. As seen in the table, the multilayer perceptron and the decision tree provide the best results, for an overall average F-measure of 27%. Interestingly, the results obtained when the number of keywords is automatically selected by the learning method (decision-based) are comparable to those when the number of keywords is selected a-priori (ratio-based), indicating the ability of the machine learning algorithm to correctly identify the correct keywords.

Additionally, we also ran an experiment to determine the amount of training data required by the system. While the learning curve continues to grow with additional amounts of data, the steepest part of the curve is observed for up to 10% of the training data, which indicates that a relatively small amount of data (about 25 books) is enough to sustain the system.

It is worth noting that the task of creating back-of-the-book indexes is highly subjective. In order to put the performance figures in perspective, one should also look at the inter-annotator agreement be-

tween human indexers as an upper bound of performance. Although we are not aware of any comprehensive studies for inter-annotator agreement on back-of-the-book indexing, we can look at the consistency studies that have been carried out on the MEDLINE corpus (Funk and Reid, 1983), where an inter-annotator agreement of 48% was found on an indexing task using a domain-specific controlled vocabulary of subject headings.

#### 4.1 Comparison with Other Systems

We compare the performance of our system with two other methods for keyword extraction. One is the *tf.idf* method, traditionally used in information retrieval as a mechanism to assign words in a text with a weight reflecting their importance. This *tf.idf* baseline system uses the same candidate extraction and filtering techniques as our supervised systems. The other baseline is the KEA keyword extraction system (Frank et al., 1999), a state-of-the-art algorithm for supervised keyword extraction. Very briefly, KEA is a supervised system that uses a Naïve Bayes learning algorithm and several features, including information theoretic features such as *tf.idf* and positional features reflecting the position of the words with respect to the beginning of the text. The KEA system was trained on the same training data set as used in our experiments.

Table 3 shows the performance obtained by these methods on the test data set. Since none of these methods have the ability to automatically determine the number of keywords to be extracted, the evaluation of these methods is done under the ratio-based setting, and thus for each method the top 0.45% ranked keywords are extracted.

Algorithm	P	R	F
<i>tf.idf</i>	8.09	8.63	8.35
KEA	11.18	11.48	11.32

Table 3: Baseline systems

#### 4.2 Performance of Individual Features

We also carried out experiments to determine the role played by each feature, by using the information gain weight as assigned by the learning algorithm. Note that ablation studies are not appropriate in our case, since the features are not orthogonal (e.g., there is high redundancy between the construction integration and the informativeness features), and thus we cannot entirely eliminate a feature from the system.

Algorithm	ratio-based			decision-based		
	P	R	F	P	R	F
Multilayer perceptron	27.98	27.77	27.87	23.93	31.98	27.38
Decision tree	27.06	27.13	27.09	22.75	34.12	27.30
SVM	20.94	20.35	20.64	21.76	30.27	25.32

Table 2: Evaluation results

Feature	Weight
part-of-speech pattern	0.1935
CI shortterm	0.1744
Wikipedia keyphraseness	0.1731
CI maxscore	0.1689
CI shortterm normalized	0.1379
ChiInformativeness	0.1122
document frequency (df)	0.1031
tf.idf	0.0870
ChiPhraseness	0.0660
length of phrase	0.0416
named entity heuristic	0.0279
within document frequency	0.0227
term frequency (tf)	0.0209

Table 4: Information gain feature weight

Table 4 shows the weight associated with each feature. Perhaps not surprisingly, the features with the highest weight are the linguistically motivated features, including syntactic patterns and the construction integration features. The Wikipedia keyphraseness also has a high score. The smallest weights belong to the information theoretic features, including term and document frequency.

## 5 Related Work

With a few exceptions (Schutze, 1998; Csomai and Mihalcea, 2007), very little work has been carried out to date on methods for *automatic* back-of-the-book index construction.

The task that is closest to ours is perhaps keyword extraction, which targets the identification of the most important words or phrases inside a document. The state-of-the-art in keyword extraction is currently represented by supervised learning methods, where a system is trained to recognize keywords in a text, based on lexical and syntactic features. This approach was first suggested in (Turney, 1999), where parameterized heuristic rules are combined with a genetic algorithm into a system for keyphrase extraction (GenEx) that automatically identifies keywords in a document. A different learning algorithm was used in (Frank et al., 1999), where a Naive Bayes learning scheme is applied on the document

collection, with improved results observed on the same data set as used in (Turney, 1999). Neither Turney nor Frank report on the recall of their systems, but only on precision: a 29.0% precision is achieved with GenEx (Turney, 1999) for five keyphrases extracted per document, and 18.3% precision achieved with Kea (Frank et al., 1999) for fifteen keyphrases per document. Finally, in recent work, (Hulth, 2003) proposes a system for keyword extraction from abstracts that uses supervised learning with lexical and syntactic features, which proved to improve significantly over previously published results.

## 6 Conclusions and Future Work

In this paper, we introduced a supervised method for back-of-the-book indexing which relies on a novel set of features, including features based on discourse comprehension, syntactic patterns, and information drawn from an online encyclopedia. According to an information gain measure of feature importance, the new features performed significantly better than the traditional frequency-based techniques, leading to a system with an F-measure of 27%. This represents an improvement of 140% with respect to a state-of-the-art supervised method for keyword extraction. Our system proved to be successful both in ranking the phrases in terms of their suitability as index entries, as well as in determining the optimal number of entries to be included in the index. Future work will focus on developing methodologies for computer-assisted back-of-the-book indexing, as well as on the use of the automatically extracted indexes in improving the browsing of digital libraries.

## Acknowledgments

We are grateful to Kirk Hastings from the California Digital Library for his help in obtaining the UC Press corpus. This research has been partially supported by a grant from Google Inc. and a grant from the Texas Advanced Research Program (#003594).

## References

- S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7).
- A. Csomai and R. Mihalcea. 2006. Creating a testbed for the evaluation of automatically generated back-of-the-book indexes. In *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing*, pages 19–25, Mexico City.
- A. Csomai and R. Mihalcea. 2007. Investigations in unsupervised back-of-the-book indexing. In *Proceedings of the Florida Artificial Intelligence Research Society*, Key West.
- D. Feng, J. Kim, E. Shaw, and E. Hovy. 2006. Towards modeling threaded discussions through ontology-based analysis. In *Proceedings of National Conference on Artificial Intelligence*.
- E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*.
- M. E. Funk and C.A. Reid. 1983. Indexing consistency in medline. *Bulletin of the Medical Library Association*, 71(2).
- A. Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Japan, August.
- W. Kintsch. 1998. *Comprehension: A paradigm for cognition*. Cambridge University Press.
- M. Kubat and S. Matwin. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In *Proceedings of the 14th International Conference on Machine Learning*.
- T. K. Landauer, P. Foltz, and D. Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes*, 25.
- C. Manning and H. Schutze. 1999. *Foundations of Natural Language Processing*. MIT Press.
- O. Medelyan and I. H. Witten. 2006. Thesaurus based automatic keyphrase indexing. In *Proceedings of the Joint Conference on Digital Libraries*.
- M. Munoz, V. Punyakanok, D. Roth, and D. Zimak. 1999. A learning approach to shallow parsing. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing*.
- P. Pecina and P. Schlesinger. 2006. Combining association measures for collocation extraction. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 651–658, Sydney, Australia.
- G. Salton and C. Buckley. 1997. Term weighting approaches in automatic text retrieval. In *Readings in Information Retrieval*. Morgan Kaufmann Publishers, San Francisco, CA.
- H. Schutze. 1998. The hypertext concordance: a better back-of-the-book index. In *Proceedings of Computerm*, pages 101–104.
- P. Turney and M. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 4(21):315–346.
- P. Turney. 1999. Learning to extract keyphrases from text. Technical report, National Research Council, Institute for Information Technology.
- G. Weiss and F. Provost. 2001. The effect of class distribution on classifier learning. Technical Report ML-TR 43, Rutgers University.



# Resolving Personal Names in Email Using Context Expansion

Tamer Elsayed,\* Douglas W. Oard,<sup>†</sup> and Galileo Namata\*

Human Language Technology Center of Excellence and  
UMIACS Laboratory for Computational Linguistics and Information Processing (CLIP)  
University of Maryland, College Park, MD 20742  
{telsayed, oard, gnamata}@umd.edu

## Abstract

This paper describes a computational approach to resolving the true referent of a named mention of a person in the body of an email. A generative model of mention generation is used to guide mention resolution. Results on three relatively small collections indicate that the accuracy of this approach compares favorably to the best known techniques, and results on the full CMU Enron collection indicate that it scales well to larger collections.

## 1 Introduction

The increasing prevalence of informal text from which a dialog structure can be reconstructed (e.g., email or instant messaging), raises new challenges if we are to help users make sense of this cacophony. Large collections offer greater scope for assembling evidence to help with that task, but they pose additional challenges as well. With well over 100,000 unique email addresses in the CMU version of the Enron collection (Klimt and Yang, 2004), common names (e.g., John) might easily refer to any one of several hundred people. In this paper, we associate named mentions in unstructured text (i.e., the body of an email and/or the subject line) to modeled identities. We see at least two direct applications for this work: (1) helping searchers who are unfamiliar with the contents of an email collection (e.g., historians or lawyers) better understand the context of emails that they find, and (2) augmenting more typical social networks (based on senders and recipients) with additional links based on references found in unstructured text.

Most approaches to resolving identity can be decomposed into four sub-problems: (1) finding a reference that requires resolution, (2) identifying candidates, (3) assembling evidence, and (4) choosing

among the candidates based on the evidence. For the work reported in this paper, we rely on the user to designate references requiring resolution (which we model as a predetermined set of mention-queries for which the correct referent is known). Candidate identification is a computational expedient that permits the evidence assembly effort to be efficiently focused; we use only simple techniques for that task. Our principal contributions are the approaches we take to evidence generation (leveraging three ways of linking to other emails where evidence might be found: reply chains, social interaction, and topical similarity) and our approach to choosing among candidates (based on a generative model of reference production). We evaluate the effectiveness of our approach on four collections, three of which have previously reported results for comparison, and one that is considerably larger than the others.

The remainder of this paper is as follows. Section 2 surveys prior work. Section 3 then describes our approach to modeling identity and ranking candidates. Section 4 presents results, and Section 5 concludes.

## 2 Related Work

The problem of identity resolution in email is a special case of the more general problem referred to as “Entity Resolution.” Entity resolution is generically defined as a process of determining the mapping from references (e.g., names, phrases) observed in data to real-world entities (e.g., persons, locations). In our case, the problem is to map *mentions* in emails to the *identities* of the individuals being referred to.

Various approaches have been proposed for entity resolution. In structured data (e.g., databases), approaches have included minimizing the number of “matching” and “merging” operations (Benjelloun et al., 2006), using global relational information (Malin, 2005; Bhattacharya and Getoor, 2007; Reuther, 2006) and using a probabilistic generative

\*Department of Computer Science

<sup>†</sup>College of Information Studies

model (Bhattacharya and Getoor, 2006). None of these approaches, however, both make use of conversational, topical, and time aspects, shown important in resolving personal names (Reuther, 2006), and take into account global relational information. Similarly, approaches in unstructured data (e.g., text) have involved using clustering techniques over biographical facts (Mann and Yarowsky, 2003), within-document resolution (Blume, 2005), and discriminative unsupervised generative models (Li et al., 2005). These too are insufficient for our problem since they suffer from inability scale or to handle early negotiation.

Specific to the problem of resolving mentions in email collections, Abadi (Abadi, 2003) used email orders from an online retailer to resolve product mentions in orders and Holzer et al. (Holzer et al., 2005) used the Web to acquire information about individuals mentioned in headers of an email collection. Our work is focused on resolving personal name references in the full email including the message body; a problem first explored by Diehl et al. (Diehl et al., 2006) using header-based traffic analysis techniques. Minkov et al. (Minkov et al., 2006) studied the same problem using a lazy graph walk based on both headers and content. Those two recent studies reported results on different test collections, however, making direct comparisons difficult. We have therefore adopted their test collections in order to establish a common point of reference.

### 3 Mention Resolution Approach

The problem we are interested in is the resolution of a personal-name mention (i.e., a named reference to a person)  $m$ , in a specific email  $e^m$  in the given collection of emails  $E$ , to its true referent. We assume that the user will designate such mention. This can be formulated as a *known-item* retrieval problem (Allen, 1989) since there is always only one right answer. Our goal is to develop a system that provides a list of potential candidates, ranked according to how strongly the system believes that a candidate is the true referent meant by the email author. In this paper, we propose a probabilistic approach that ranks the candidates based on the estimated probability of having been mentioned. Formally, we seek to estimate the probability  $p(c|m)$  that a potential candi-

date  $c$  is the one referred to by the given mention  $m$ , over all candidates  $C$ .

We define a mention  $m$  as a tuple  $\langle l^m, e^m \rangle$ , where  $l^m$  is the “literal” string of characters that represents  $m$  and  $e^m$  is the email where  $m$  is observed.<sup>1</sup> We assume that  $m$  can be resolved to a distinguishable participant for whom at least one email address is present in the collection.<sup>2</sup>

The probabilistic approach we propose is motivated by a generative scenario of mentioning people in email. The scenario begins with the author of the email  $e^m$ , intending to refer to a person in that email. To do that s/he will:

1. Select a person  $c$  to whom s/he will refer
2. Select an appropriate context  $x_k$  to mention  $c$
3. Select a specific lexical reference  $l^m$  to refer to  $c$  given the context  $x_k$ .

For example, suppose “John” is sending an email to “Steve” and wants to mention a common friend “Edward.” “John” knows that he and Steve know 2 people named Edward, one is a friend of both known by “Ed” and the other is his soccer trainer. If “John” would like to talk about the former, he would use “Ed” but he would likely use “Edward” plus some terms (e.g., “soccer”, “team”, etc) for the latter. “John” relies on the social context, or the topical context, for “Steve” to disambiguate the mention.

The steps of this scenario impose a certain structure to our solution. First, we need to have a representational model for each candidate identity. Second, we need to reconstruct the context of the queried mention. Third, it requires a computational model of identity that supports reasoning about identities. Finally, it requires a resolution technique that leverages both the identity models and the context to rank the potential candidates. In this section, we will present our resolution approach within that structure. We first discuss how to build both representational and computational models of identity in section 3.1. Next, we introduce a definition of the contextual space and how we can reconstruct it in

<sup>1</sup>The exact position in  $e^m$  where  $l^m$  is observed should also be included in the definition, but we ignore it assuming that all matched literal mentions in one email refer to the same identity.

<sup>2</sup>Resolving mentions that refer to non-participants is outside the scope of this paper.

section 3.2. Finally, we link those pieces together by the resolution algorithm in section 3.3.

### 3.1 Computational Model of Identity

**Representation:** In a collection of emails, individuals often use different email addresses, multiple forms of their proper names, and different nicknames. In order to track references to a person over a large collection, we need to capture as many as possible of these *referential* attributes in one representation. We extend our simple representation of identity proposed in (Elsayed and Oard, 2006) where an identity is represented by a set of pairwise co-occurrence of referential attributes (i.e., co-occurrence “associations”), and each extracted association has a frequency of occurrence. The attributes are extracted from the headers and salutation and signature lines. For example, an “address-nickname” association  $\langle a, n \rangle$  is inferred whenever a nickname  $n$  is usually observed in signature lines of emails sent from email address  $a$ . Three types of referential attributes were identified in the original representation: email addresses, names, and nicknames. We add usernames as well to account for the absence of any other type of names. Names, nicknames, and usernames are distinguishable based on where each is extracted: email addresses and names from headers, nicknames from salutation and signature lines, and usernames from email addresses. Since (except in rare cases) an email address is bound to one personal identity, the model leverages email addresses as the basis by mandating that at least one email address must appear in any observed association. As an off-line preprocessing step, we extract the referential attributes from the whole collection and build the identity models. The first step in the resolution process is to determine the list of identity models that are viable candidates as the true referent. For the experiments reported in this paper, any identity model with a first name or nickname that exactly matches the mention is considered a candidate.

**Labeling Observed Names:** For the purpose of resolving name mentions, it is necessary to compute the probability  $p(l|c)$  that a person  $c$  is referred to by a given “literal” mention  $l$ . Intuitively, that probability can be estimated based on the observed “name-type” of  $l$  and how often that association occurs in

the represented model. We define  $T$  as the set of 3 different types of single-token name-types: first, last, and nickname. We did not handle middle names and initials, just for simplicity. Names that are extracted from salutation and signature lines are labeled as nicknames whereas full names extracted from headers are first normalized to “First Last” form and then each single token is labeled based on its relative position as being the first or last name. Usernames are treated similarly to full names if they have more than one token, otherwise they are ignored. Note that the same single-token name may appear as a first name and a nickname.

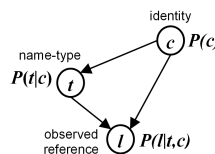


Figure 1: A computational model of identity.

**Reasoning:** Having tokenized and labeled all names, we propose to model the association of a single-token name  $l$  of type  $t$  to an identity  $c$  by a simple 3-node Bayesian network illustrated in Figure 1. In the network, the observed mention  $l$  is distributed conditionally on both the identity  $c$  and the name-type  $t$ .  $p(c)$  is the prior probability of observing the identity  $c$  in the collection.  $p(t|c)$  is the probability that a name-type  $t$  is used to refer to  $c$ .  $p(l|t, c)$  is the probability of referring to  $c$  by  $l$  of type  $t$ . These probabilities can be inferred from the representational model as follows:

$$p(c) = \frac{|assoc(c)|}{\sum_{c' \in C} |assoc(c')|}$$

$$p(t|c) = \frac{freq(t, c)}{\sum_{t' \in T} freq(t', c)}$$

$$p(l|t, c) = \frac{freq(l, t, c)}{\sum_{l' \in assoc(c)} freq(l', t, c)}$$

where  $assoc(c)$  is the set of observed associations of referential attributes in the represented model  $c$ .

The probability of observing a mention  $l$  given that it belongs to an identity  $c$ , without assuming a specific token type, can then be inferred as follows:

$$p(l|c) = \sum_{t \in T} p(t|c) p(l|t, c)$$

In the case of a multi-token names (e.g., John Smith), we assume that the first is either a first name

or nickname and the last is a last name, and compute it accordingly as follows:

$$p(l_1 l_2 | c) = \left\{ \sum_{t \in \{f, n\}} p(t | c) p(l_1 | t, c) \right\} \cdot p(l_2 | last, c)$$

where  $f$  and  $n$  above denotes first name and nickname respectively.

Email addresses are also handled, but in a different way. Since we assume each of them uniquely identifies the identity, all email addresses for one identity are mapped to just one of them, which then has half of the probability mass (because it appears in every extracted co-occurrence association).

Our computational model of identity can be thought of as a language model over a set of personal references and thus it is important to account for unobserved references. If we know that a specific first name often has a common nickname (by a dictionary of commonly used first to nickname mappings (e.g., Robert to Bob)), but this nickname was not observed in the corpus, we will need to apply smoothing. We achieve that by assuming the nickname would have been observed  $n$  times where  $n$  is some fraction (0.75 in our experiments) of the frequency of the observed name. We repeat that for each unobserved nickname and then treat them as if they were actually observed.

## 3.2 Contextual Space

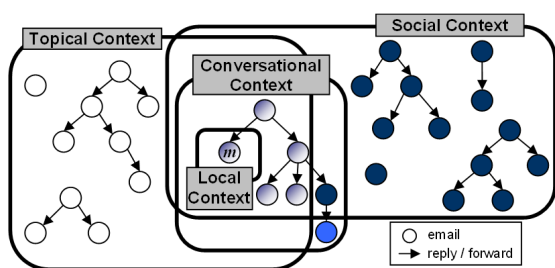


Figure 2: Contextual Space

It is obvious that understanding the context of an ambiguous mention will help with resolving it.

Fortunately, the nature of email as a conversational medium and the link-relationships between emails and people over time can reveal clues that can be exploited to partially reconstruct that context.

We define the contextual space  $X(m)$  of a mention  $m$  as a *mixture* of 4 types of contexts with  $\lambda_k$  as the mixing coefficient of context  $x_k$ . The four contexts (illustrated in Figure 2) are:

(1) **Local Context:** the email  $e^m$  where the named person is mentioned.

(2) **Conversational Context:** emails in the broader discussion that includes  $e^m$ , typically the thread that contains it.

(3) **Social Context:** discussions that some or all of the participants (sender and receivers) of  $e^m$  joined or initiated at around the time of the mention-email. These might bear some otherwise-undetected relationship to the mention-email.

(4) **Topical Context:** discussions that are topically similar to the mention-discussion that took place at around the time of  $e^m$ , regardless of whether the discussions share any common participants.

These generally represent a growing (although not strictly nested) contextual space around the queried mention. We assume that all mentions in an email share the same contextual space. Therefore, we can treat the context of a mention as the context of its email. However, each email in the collection has its own contextual space that could overlap with another email's space.

### 3.2.1 Formal Definition

We define  $K$  as the set of the 4 types of contexts. A context  $x_k$  is represented by a probability distribution over all emails in the collection. An email  $e_j$  belongs to the  $k^{\text{th}}$  context of another email  $e_i$  with probability  $p(e_j | x_k(e_i))$ . How we actually represent each context and estimate the distribution depends upon the type of the context. We explain that in detail in section 3.2.2.

### 3.2.2 Context Reconstruction

In this section, we describe how each context is constructed.

**Local Context:** Since this is simply  $e^m$ , all of the probability mass is assigned to it.

**Conversational Context:** Threads (i.e., reply chains) are imperfect approximations of focused discussions, since people sometimes switch topics within a thread (and indeed sometimes within the same email). We nonetheless expect threads to exhibit a useful degree of focus and we have therefore adopted them as a computational representation of a discussion in our experiments. To reconstruct threads in the collection, we adopted the technique introduced in (Lewis and Knowles, 1997). Thread

reconstruction results in a unique tree containing the mention-email. Although we can distinguish between different paths or subtrees of that tree, we elected to have a uniform distribution over all emails in the same thread. This also applies to threads retrieved in the social and topical contexts as well.

**Social Context:** Discussions that share common participants may also be useful, though we expect their utility to decay somewhat with time. To reconstruct that context, we temporally rank emails that share at least one participant with  $e^m$  in a time period around  $e^m$  and then expand each by its thread (with duplicate removal). Emails in each thread are then each assigned a weight that equals the reciprocal of its thread rank. We do that separately for emails that temporally precede or follow  $e^m$ . Finally, weights are normalized to produce one distribution for the whole social context.

**Topical Context:** Identifying topically-similar content is a traditional query-by-example problem that has been well researched in, for example, the TREC routing task (Lewis, 1996) and the Topic Detection and Tracking evaluations (Allan, 2002). Individual emails may be quite terse, but we can exploit the conversational structure to obtain topically related text. In our experiments, we tracked back to the root of the thread in which  $e^m$  was found and used the subject line and the body text of that root email as a query to Lucene<sup>3</sup> to identify topically-similar emails. Terms found in the subject line are doubled in the query to emphasize what is sometimes a concise description of the original topic. Subsequent processing is then similar to that used for the social context, except that the emails are first ranked by their topical, rather than temporal, similarity.

The approaches we adopted to reconstruct the social and topical contexts were chosen for their relative simplicity, but there are clearly more sophisticated alternatives. For example, topic modeling techniques (McCallum et al., 2005) could be leveraged in the reconstruction of the topical context.

### 3.3 Mention Resolution

Given a specific mention  $m$  and the set of identity models  $C$ , our goal now is to compute  $p(c|m)$  for each candidate  $c$  and rank them accordingly.

<sup>3</sup><http://lucene.apache.org>

#### 3.3.1 Context-Free Mention Resolution

If we resolve  $m$  out of its context, then we can compute  $p(c|m)$  by applying Bayes' rule as follows:

$$p(c|m) \approx p(c|l^m) = \frac{p(l^m|c) p(c)}{\sum_{c' \in C} p(l^m|c') p(c')}$$

All the terms above are estimated as discussed earlier in section 3.1. We call this approach "backoff" since it can be used as a fall-back strategy. It is considered the baseline approach in our experiments.

#### 3.3.2 Contextual Mention Resolution

We now discuss the more realistic situation in which we use the context to resolve  $m$ . By expanding the mention with its context, we get

$$p(c|m) = p(c|l^m, X(e^m))$$

We then apply Bayes' rule to get

$$p(c|l^m, X(e^m)) = \frac{p(c, l^m, X(e^m))}{p(l^m, X(e^m))}$$

where  $p(l^m, X(e^m))$  is the probability of observing  $l^m$  in the context. We can ignore this probability since it is constant across all candidates in our ranking. We now restrict our focus to the numerator  $p(c, l^m, X(e^m))$ , that is the probability that the sender chose to refer to  $c$  by  $l^m$  in the contextual space. As we discussed in section 3.2,  $X$  is defined as a mixture of contexts therefore we can further expand it as follows:

$$p(c, l^m, X(e^m)) = \sum_k \lambda_k p(c, l^m, x_k(e^m))$$

Following the intuitive generative scenario we introduced earlier, the context-specific probability can be decomposed as follows:

$$\begin{aligned} p(c, l^m, x_k(e^m)) &= p(c) \\ &\quad * p(x_k(e^m)|c) \\ &\quad * p(l^m|x_k(e^m), c) \end{aligned}$$

where  $p(c)$  is the probability of selecting a candidate  $c$ ,  $p(x_k(e^m)|c)$  is the probability of selecting  $x_k$  as an appropriate context to mention  $c$ , and  $p(l^m|x_k(e^m), c)$  is the probability of choosing to mention  $c$  by  $l^m$  given that  $x_k$  is the appropriate context.

**Choosing person to mention:**  $p(c)$  can be estimated as discussed in section 3.1.

**Choosing appropriate context:** By applying Bayes' rule to compute  $p(x_k(e^m)|c)$  we get

$$p(x_k(e^m)|c) = \frac{p(c|x_k(e^m)) p(x_k(e^m))}{p(c)}$$

$p(x_k(e^m))$  is the probability of choosing  $x_k$  to generally mention people. In our experiments, we assumed a uniform distribution over all contexts.  $p(c|x_k(e^m))$  is the probability of mentioning  $c$  in  $x_k(e^m)$ . Given that the context is defined as a distribution over emails, this can be expanded to

$$p(c|x_k(e^m)) = \sum_{e_i \in E} p(e_i|x_k(e^m)) p(c|e_i)$$

where  $p(c|e_i)$  is the probability that  $c$  is mentioned in the email  $e_i$ . This, in turn, can be estimated using the probability of referring to  $c$  by at least one unique reference observed in that email. By assuming that all lexical matches in the same email refer to the same person, and that all lexically-unique references are statistically independent, we can compute that probability as follows:

$$\begin{aligned} p(c|e_i) &= 1 - p(c \text{ is not mentioned in } e_i) \\ &= 1 - \prod_{m' \in M(e_i)} (1 - p(c|m')) \end{aligned}$$

where  $p(c|m')$  is the probability that  $c$  is the true referent of  $m'$ . This is the same general problem of resolving mentions, but now concerning a related mention  $m'$  found in the context of  $m$ . To handle this, there are two alternative solutions: (1) break the cycle and compute context-free resolution probabilities for those related mentions, or (2) jointly resolve all mentions. In this paper, we will only consider the first, leaving joint resolution for future work.

**Choosing a name-mention:** To estimate  $p(l^m|x_k(e^m), c)$ , we suggest that the email author would choose either to select a reference (or a modified version of a reference) that was previously mentioned in the context or just ignore the context. Hence, we estimate that probability as follows:

$$\begin{aligned} p(l^m|x_k(e^m), c) &= \alpha p(l^m \in x_k(e^m)|c) \\ &\quad + (1 - \alpha) p(l^m|c) \end{aligned}$$

where  $\alpha \in [0, 1]$  is a mixing parameter (set at 0.9 in our experiments), and  $p(l^m|c)$  is estimated as in section 3.1.  $p(l^m \in x_k(e^m)|c)$  can be estimated as follows:

$$\begin{aligned} p(l^m \in x_k(e^m)|c) &= \\ \sum_{m' \in x_k} p(l^m|l^{m'}) p(l^{m'}|x_k) p(c|l^{m'}) \end{aligned}$$

where  $p(l^m|l^{m'})$  is the probability of modifying  $l^{m'}$  into  $l^m$ . We assume all possible mentions of  $c$

are equally similar to  $m$  and estimate  $p(l^m|l^{m'})$  by  $\frac{1}{|\text{possible mentions of } c|} \cdot p(l^{m'}|x_k)$  is the probability of observing  $l^{m'}$  in  $x_k$ , which we estimate by its relative frequency in that context. Finally,  $p(c|l^{m'})$  is again a mention resolution problem concerning the reference  $r_i$  which can be resolved as shown earlier.

The Aho-Corasick linear-time algorithm (Aho and Corasick, 1975) is used to find mentions of names, using a corpus-based dictionary that includes all names, nicknames, and email addresses extracted in the preprocessing step.

## 4 Experimental Evaluation

We evaluate our mention resolution approach using four test collections, all are based on the CMU version of the Enron collection; each was created by selecting a subset of that collection, selecting a set of query-mentions within emails from that subset, and creating an answer key in which each query-mention is associated with a single email address.

The first two test collections were created by Minkov et al (Minkov et al., 2006). These test collections correspond to two email accounts, “sager-e” (the “Sager” collection) and “shapiro-r” (the “Shapiro” collection). Their mention-queries and answer keys were generated automatically by identifying name mentions that correspond uniquely to individuals referenced in the cc header, and eliminating that cc entry from the header.

The third test collection, which we call the “Enron-subset” is an extended version of the test collection created by Diehl et al (Diehl et al., 2006). Emails from all top-level folders were included in the collection, but only those that were both sent by and received by at least one email address of the form <name1>.<name2>@enron.com were retained. A set of 78 mention-queries were manually selected and manually associated with the email address of the true referent by the third author using an interactive search system developed specifically to support that task. The set of queries was limited to those that resolve to an address of the form <name1>.<name2>@enron.com. Names found in salutation or signature lines or that exactly match <name1> or <name2> of any of the email participants were not selected as query-mentions. Those 78 queries include the 54 used by Diehl et al.

Table 1: Test collections used in the experiments.

Test Coll.	Emails	IDs	Queries	Candidates
Sager	1,628	627	51	4 (1-11)
Shapiro	974	855	49	8 (1-21)
Enron-sub	54,018	27,340	78	152 (1-489)
Enron-all	248,451	123,783	78	518 (3-1785)

For our fourth test collection (“Enron-all”), we used the same 78 mention-queries and the answer key from the Enron-subset collection, but we used the full CMU version of the Enron collection (with duplicates removed). We use this collection to assess the scalability of our techniques.

Some descriptive statistics for each test collection are shown in Table 1. The Sager and Shapiro collections are typical of personal collections, while the other two represent organizational collections. These two types of collections differ markedly in the number of known identities and the candidate list sizes as shown in the table (the candidate list size is presented as an average over that collection’s mention-queries and as the full range of values).

#### 4.1 Evaluation Measures

There are two commonly used single-valued evaluation measures for “known item”-retrieval tasks. The “*Success @ 1*” measure characterizes the accuracy of one-best selection, computed as the mean across queries of the precision at the top rank for each query. For a single-valued figure of merit that considers every list position, we use “*Mean Reciprocal Rank*” (MRR), computed as the mean across queries of the inverse of the rank at which the correct referent is found.

#### 4.2 Results

There are four basic questions which we address in our experimental evaluation: (1) How does our approach perform compared to other approaches?, (2) How is it affected by the size of the collection and by increasing the time period?, (3) Which context makes the most important contribution to the resolution task? and (4) Does the mixture help?

In our experiments, we set the mixing coefficients  $\lambda_k$  and the context priors  $p(x_k)$  to a uniform distribution over all reconstructed contexts.

To compare our system performance with results

Table 2: Accuracy results with different time periods.

	Period (days)	MRR		Success @ 1	
		Prob.	Minkov	Prob.	Minkov
Sager	10	0.899	0.889	0.843	0.804
	100	<b>0.911</b>	0.889	<b>0.863</b>	0.804
	200	<b>0.911</b>	0.889	<b>0.863</b>	0.804
Shapiro	10	<b>0.913</b>	0.879	0.857	0.779
	100	0.910	0.879	0.837	0.779
	200	0.911	0.837	<b>0.878</b>	0.779
Enron-sub	10	0.878	-	0.821	-
	100	<b>0.911</b>	-	<b>0.846</b>	-
	200	<b>0.911</b>	-	<b>0.846</b>	-
Enron-all	10	<b>0.890</b>	-	<b>0.821</b>	-
	100	0.888	-	<b>0.821</b>	-
	200	0.888	-	<b>0.821</b>	-

previously reported, we experimented with different (symmetric) time periods for selecting threads in the social and topical contexts. Three representative time periods, in days, were arbitrarily chosen: 10 (i.e., +/- 5) days, 100 (i.e., +/- 50) days, and 200 (i.e., +/- 100) days. In each case, the mention-email defines the center of this period.

A summary of our results (denoted by “Prob.”) are shown in Table 2 with the best results for each test collection highlighted in bold. The table also includes the results reported in Minkov et al (Minkov et al., 2006) for the small collections for comparison purposes.<sup>4</sup> Each score for our system was the best over all combinations of contexts for these collections and time periods. Given these scores, our results compare favorably with the previously reported results for both Sager and Shapiro collections.

Another notable thing about our results is that they seem to be good enough for practical applications. Specifically, our one-best selection (over all tried conditions) is correct at least 82% of the time over all collections, including the largest one. Of course, the Enron-focused selection of mention-queries in every case is an important caveat on these results; we do not yet know how well our techniques will hold up with less evidence, as might be the case for mentions of people from outside Enron.

It is encouraging that testing on the largest col-

<sup>4</sup>For the “Enron-subset” collection, we do not know which 54 mention-queries Diehl et al used in (Diehl et al., 2006)

lection (with all unrelated and thus noisy data) did not hurt the effectiveness much. For the three different time periods we tried, there was no systematic effect.

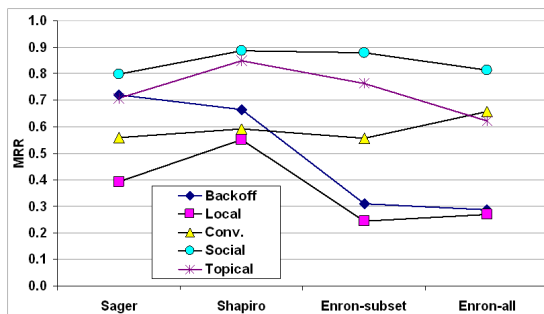


Figure 3: Individual contexts, period set to 100 days.

**Individual Contexts:** Our choice of contexts was motivated by intuition rather than experiments, so we also took this opportunity to characterize the contribution of each context to the results. We did that by setting some of the context mixing-coefficients to zero and leaving the others equally-weighted. Figure 3 shows the MRR achieved with each context. In that figure, the “backoff” curve indicates how well the simple context-free resolution would do. The difference between the two smallest and the two largest collections is immediately apparent—this backoff is remarkably effective for the smaller collections, and almost useless for the larger ones, suggesting that the two smaller collections are essentially much easier. The social context is clearly quite useful, more so than any other single context, for every collection. This tends to support our expectation that social networks can be as informative as content networks in email collections. The topical context also seems to be useful on its own. The conversational context is moderately useful on its own in the larger collections. The local context alone is not very informative for the larger collections.

**Mixture of Contexts:** The principal motivation for combining different types of contexts is that different sources may provide complementary evidence. To characterize that effect, we look at combinations of contexts. Figure 4 shows three such context combinations, anchored by the social context alone, with a 100-day window (the results for 10 and 200 day periods are similar). Reassuringly, adding more contexts (hence more evidence) turns out to be a rea-

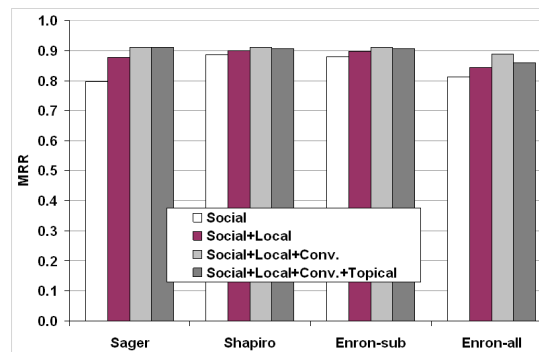


Figure 4: Mixture of contexts, period set to 100 days.

sonable choice in most cases. For the full combination, we notice a drop in the effectiveness from the addition of the topical context.<sup>5</sup> This suggests that the construction of the topical context may need more careful design, and/or that learned  $\lambda_k$ 's could yield better evidence combination (since these results were obtained with equal  $\lambda_k$ 's).

## 5 Conclusion

We have presented an approach to mention resolution in email that flexibly makes use of expanding contexts to accurately resolve the identity of a given mention. Our approach focuses on four naturally occurring contexts in email, including a message, a thread, other emails with senders and/or recipients in common, and other emails with significant topical content in common. Our approach outperforms previously reported techniques and it scales well to larger collections. Moreover, our results serve to highlight the importance of social context when resolving mentions in social media, which is an idea that deserves more attention generally. In future work, we plan to extend our test collection with mention queries that must be resolved in the “long tail” of the identity distribution where less evidence is available. We are also interested in exploring iterative approaches to jointly resolving mentions.

## Acknowledgments

The authors would like to thank Lise Getoor for her helpful advice.

<sup>5</sup>This also occurs even when topical context is combined with only social context.



## References

- Daniel J. Abadi. 2003. Comparing domain-specific and non-domain-specific anaphora resolution techniques. Cambridge University MPhil Dissertation.
- Alfred V. Aho and Margaret J. Corasick. 1975. Efficient string matching: an aid to bibliographic search. In *Communications of the ACM*.
- James Allan, editor. 2002. *Topic detection and tracking: event-based information organization*. Kluwer Academic Publishers, Norwell, MA, USA.
- Bryce Allen. 1989. Recall cues in known-item retrieval. *JASIS*, 40(4):246–252.
- Omar Benjelloun, Hector Garcia-Molina, Hideki Kawai, Tait Elliott Larson, David Menestrina, Qi Su, Sutthipong Thavisomboon, and Jennifer Widom. 2006. Generic entity resolution in the serf project. *IEEE Data Engineering Bulletin*, June.
- Indrajit Bhattacharya and Lise Getoor. 2006. A latent dirichlet model for unsupervised entity resolution. In *The SIAM International Conference on Data Mining (SIAM-SDM)*, Bethesda, MD, USA.
- Indrajit Bhattacharya and Lise Getoor. 2007. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1), March.
- Matthias Blume. 2005. Automatic entity disambiguation: Benefits to NER, relation extraction, link analysis, and inference. In *International Conference on Intelligence Analysis*, May.
- Chris Diehl, Lise Getoor, and Galileo Namata. 2006. Name reference resolution in organizational email archives. In *Proceedings of SIAM International Conference on Data Mining*, Bethesda, MD, USA, April 20–22.
- Tamer Elsayed and Douglas W. Oard. 2006. Modeling identity in archival collections of email: A preliminary study. In *Proceedings of the 2006 Conference on Email and Anti-Spam (CEAS 06)*, pages 95–103, Mountain View, California, July.
- Ralf Holzer, Bradley Malin, and Latanya Sweeney. 2005. Email alias detection using social network analysis. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 52–57, New York, NY, USA. ACM Press.
- Bryan Klimt and Yiming Yang. 2004. Introducing the Enron corpus. In *Conference on Email and Anti-Spam*, Mountain view, CA, USA, July 30–31.
- David D. Lewis and Kimberly A. Knowles. 1997. Threading electronic mail: a preliminary study. *Inf. Process. Manage.*, 33(2):209–217.
- David D. Lewis. 1996. The trec-4 filtering track. In *The Fourth Text REtrieval Conference (TREC-4)*, pages 165–180, Gaithersburg, Maryland.
- Xin Li, Paul Morie, and Dan Roth. 2005. Semantic integration in text: from ambiguous names to identifiable entities. *AI Magazine. Special Issue on Semantic Integration*, 26(1):45–58.
- Bradley Malin. 2005. Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counter-terrorism, and Security, in conjunction with the SIAM International Conference on Data Mining*, Newport Beach, CA, USA, April 21–23.
- Gideon S. Mann and David Yarowsky. 2003. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 33–40, Morristown, NJ, USA. Association for Computational Linguistics.
- Andrew McCallum, Andres Corrada-Emmanuel, and Xuerui Wang Wang. 2005. Topic and role discovery in social networks. In *IJCAI*.
- Einat Minkov, William W. Cohen, and Andrew Y. Ng. 2006. Contextual search and name disambiguation in email using graphs. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 27–34, New York, NY, USA. ACM Press.
- Patric Reuther. 2006. Personal name matching: New test collections and a social network based approach.

# Integrating Graph-Based and Transition-Based Dependency Parsers

Joakim Nivre

Växjö University      Uppsala University  
Computer Science    Linguistics and Philology  
SE-35195 Växjö      SE-75126 Uppsala  
nivre@msi.vxu.se

Ryan McDonald

Google Inc.  
76 Ninth Avenue  
New York, NY 10011  
ryanmcd@google.com

## Abstract

Previous studies of data-driven dependency parsing have shown that the distribution of parsing errors are correlated with theoretical properties of the models used for learning and inference. In this paper, we show how these results can be exploited to improve parsing accuracy by integrating a graph-based and a transition-based model. By letting one model generate features for the other, we consistently improve accuracy for both models, resulting in a significant improvement of the state of the art when evaluated on data sets from the CoNLL-X shared task.

## 1 Introduction

Syntactic dependency graphs have recently gained a wide interest in the natural language processing community and have been used for many problems ranging from machine translation (Ding and Palmer, 2004) to ontology construction (Snow et al., 2005). A dependency graph for a sentence represents each word and its syntactic dependents through labeled directed arcs, as shown in figure 1. One advantage of this representation is that it extends naturally to discontinuous constructions, which arise due to long distance dependencies or in languages where syntactic structure is encoded in morphology rather than in word order. This is undoubtedly one of the reasons for the emergence of dependency parsers for a wide range of languages. Many of these parsers are based on data-driven parsing models, which learn to produce dependency graphs for sentences solely from an annotated corpus and can be easily ported to any

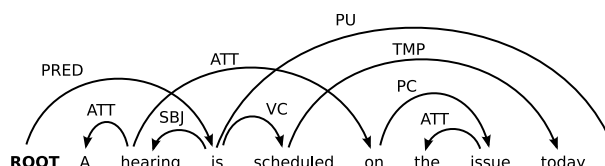


Figure 1: Dependency graph for an English sentence.

language or domain in which annotated resources exist.

Practically all data-driven models that have been proposed for dependency parsing in recent years can be described as either *graph-based* or *transition-based* (McDonald and Nivre, 2007). In graph-based parsing, we learn a model for scoring possible dependency graphs for a given sentence, typically by factoring the graphs into their component arcs, and perform parsing by searching for the highest-scoring graph. This type of model has been used by, among others, Eisner (1996), McDonald et al. (2005a), and Nakagawa (2007). In transition-based parsing, we instead learn a model for scoring transitions from one parser state to the next, conditioned on the parse history, and perform parsing by greedily taking the highest-scoring transition out of every parser state until we have derived a complete dependency graph. This approach is represented, for example, by the models of Yamada and Matsumoto (2003), Nivre et al. (2004), and Attardi (2006).

Theoretically, these approaches are very different. The graph-based models are globally trained and use exact inference algorithms, but define features over a limited history of parsing decisions. The transition-based models are essentially the opposite. They use local training and greedy inference algorithms, but

define features over a rich history of parsing decisions. This is a fundamental trade-off that is hard to overcome by tractable means. Both models have been used to achieve state-of-the-art accuracy for a wide range of languages, as shown in the CoNLL shared tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007), but McDonald and Nivre (2007) showed that a detailed error analysis reveals important differences in the distribution of errors associated with the two models.

In this paper, we consider a simple way of integrating graph-based and transition-based models in order to exploit their complementary strengths and thereby improve parsing accuracy beyond what is possible by either model in isolation. The method integrates the two models by allowing the output of one model to define features for the other. This method is simple – requiring only the definition of new features – and robust by allowing a model to learn relative to the predictions of the other.

## 2 Two Models for Dependency Parsing

### 2.1 Preliminaries

Given a set  $L = \{l_1, \dots, l_{|L|}\}$  of arc labels (dependency relations), a dependency graph for an input sentence  $x = w_0, w_1, \dots, w_n$  (where  $w_0 = \text{ROOT}$ ) is a labeled directed graph  $G = (V, A)$  consisting of a set of nodes  $V = \{0, 1, \dots, n\}$ <sup>1</sup> and a set of labeled directed arcs  $A \subseteq V \times V \times L$ , i.e., if  $(i, j, l) \in A$  for  $i, j \in V$  and  $l \in L$ , then there is an arc from node  $i$  to node  $j$  with label  $l$  in the graph. A dependency graph  $G$  for a sentence  $x$  must be a directed tree originating out of the root node 0 and spanning all nodes in  $V$ , as exemplified by the graph in figure 1. This is a common constraint in many dependency parsing theories and their implementations.

### 2.2 Graph-Based Models

Graph-based dependency parsers parameterize a model over smaller substructures in order to search the space of valid dependency graphs and produce the most likely one. The simplest parameterization is the arc-factored model that defines a real-valued score function for arcs  $s(i, j, l)$  and further defines the score of a dependency graph as the sum of the

score of all the arcs it contains. As a result, the dependency parsing problem is written:

$$G = \arg \max_{G=(V,A)} \sum_{(i,j,l) \in A} s(i, j, l)$$

This problem is equivalent to finding the highest scoring directed spanning tree in the complete graph over the input sentence, which can be solved in  $O(n^2)$  time (McDonald et al., 2005b). Additional parameterizations are possible that take more than one arc into account, but have varying effects on complexity (McDonald and Satta, 2007). An advantage of graph-based methods is that tractable inference enables the use of standard structured learning techniques that globally set parameters to maximize parsing performance on the training set (McDonald et al., 2005a). The primary disadvantage of these models is that scores – and as a result any feature representations – are restricted to a single arc or a small number of arcs in the graph.

The specific graph-based model studied in this work is that presented by McDonald et al. (2006), which factors scores over pairs of arcs (instead of just single arcs) and uses near exhaustive search for unlabeled parsing coupled with a separate classifier to label each arc. We call this system MSTParser, or simply MST for short, which is also the name of the freely available implementation.<sup>2</sup>

### 2.3 Transition-Based Models

Transition-based dependency parsing systems use a model parameterized over transitions of an abstract machine for deriving dependency graphs, such that every transition sequence from the designated initial configuration to some terminal configuration derives a valid dependency graph. Given a real-valued score function  $s(c, t)$  (for transition  $t$  out of configuration  $c$ ), parsing can be performed by starting from the initial configuration and taking the optimal transition  $t^* = \arg \max_{t \in T} s(c, t)$  out of every configuration  $c$  until a terminal configuration is reached. This can be seen as a greedy search for the optimal dependency graph, based on a sequence of locally optimal decisions in terms of the transition system.

Many transition systems for data-driven dependency parsing are inspired by shift-reduce parsing,

<sup>1</sup>We use the common convention of representing words by their index in the sentence.

<sup>2</sup><http://mstparser.sourceforge.net>

where each configuration  $c$  contains a stack  $\sigma_c$  for storing partially processed nodes and a buffer  $\beta_c$  containing the remaining input. Transitions in such a system add arcs to the dependency graph and manipulate the stack and buffer. One example is the transition system defined by Nivre (2003), which parses a sentence  $x = w_0, w_1, \dots, w_n$  in  $O(n)$  time.

To learn a scoring function on transitions, these systems rely on discriminative learning methods, such as memory-based learning or support vector machines, using a strictly local learning procedure where only single transitions are scored (not complete transition sequences). The main advantage of these models is that features are not restricted to a limited number of graph arcs but can take into account the entire dependency graph built so far. The major disadvantage is that the greedy parsing strategy may lead to error propagation.

The specific transition-based model studied in this work is that presented by Nivre et al. (2006), which uses support vector machines to learn transition scores. We call this system MaltParser, or Malt for short, which is also the name of the freely available implementation.<sup>3</sup>

## 2.4 Comparison and Analysis

These models differ primarily with respect to three properties: *inference*, *learning*, and *feature representation*. MaltParser uses an inference algorithm that greedily chooses the best parsing decision based on the current parser history whereas MSTParser uses exhaustive search algorithms over the space of all valid dependency graphs to find the graph that maximizes the score. MaltParser trains a model to make a single classification decision (choose the next transition) whereas MSTParser trains a model to maximize the global score of correct graphs. MaltParser can introduce a rich feature history based on previous parser decisions, whereas MSTParser is forced to restrict features to a single decision or a pair of nearby decisions in order to retain efficiency.

These differences highlight an inherent trade-off between global inference/learning and expressiveness of feature representations. MSTParser favors the former at the expense of the latter and MaltParser the opposite. This difference was highlighted in the

study of McDonald and Nivre (2007), which showed that the difference is reflected directly in the error distributions of the parsers. Thus, MaltParser is less accurate than MSTParser for long dependencies and those closer to the root of the graph, but more accurate for short dependencies and those farthest away from the root. Furthermore, MaltParser is more accurate for dependents that are nouns and pronouns, whereas MSTParser is more accurate for verbs, adjectives, adverbs, adpositions, and conjunctions.

Given that there is a strong negative correlation between dependency length and tree depth, and given that nouns and pronouns tend to be more deeply embedded than (at least) verbs and conjunctions, these patterns can all be explained by the same underlying factors. Simply put, MaltParser has an advantage in its richer feature representations, but this advantage is gradually diminished by the negative effect of error propagation due to the greedy inference strategy as sentences and dependencies get longer. MSTParser has a more even distribution of errors, which is expected given that the inference algorithm and feature representation should not prefer one type of arc over another. This naturally leads one to ask: Is it possible to integrate the two models in order to exploit their complementary strengths? This is the topic of the remainder of this paper.

## 3 Integrated Models

There are many conceivable ways of combining the two parsers, including more or less complex ensemble systems and voting schemes, which only perform the integration at parsing time. However, given that we are dealing with data-driven models, it should be possible to integrate at learning time, so that the two complementary models can learn from one another. In this paper, we propose to do this by letting one model generate features for the other.

### 3.1 Feature-Based Integration

As explained in section 2, both models essentially learn a scoring function  $s : X \rightarrow \mathbb{R}$ , where the domain  $X$  is different for the two models. For the graph-based model,  $X$  is the set of possible dependency arcs  $(i, j, l)$ ; for the transition-based model,  $X$  is the set of possible configuration-transition pairs  $(c, t)$ . But in both cases, the input is represented

<sup>3</sup><http://w3.msi.vxu.se/~jha/maltparser/>

<b>MST<sub>Malt</sub></b> – defined over $(i, j, l)$ ( $*$ = any label/node)
Is $(i, j, *)$ in $G_x^{\text{Malt}}?$
Is $(i, j, l)$ in $G_x^{\text{Malt}}?$
Is $(i, j, *)$ not in $G_x^{\text{Malt}}?$
Is $(i, j, l)$ not in $G_x^{\text{Malt}}?$
Identity of $l'$ such that $(*, j, l')$ is in $G_x^{\text{Malt}}?$
Identity of $l'$ such that $(i, j, l')$ is in $G_x^{\text{Malt}}?$
<b>Malt<sub>MST</sub></b> – defined over $(c, t)$ ( $*$ = any label/node)
Is $(\sigma_c^0, \beta_c^0, *)$ in $G_x^{\text{MST}}?$
Is $(\beta_c^0, \sigma_c^0, *)$ in $G_x^{\text{MST}}?$
Head direction for $\sigma_c^0$ in $G_x^{\text{MST}}$ (left/right/ROOT)
Head direction for $\beta_c^0$ in $G_x^{\text{MST}}$ (left/right/ROOT)
Identity of $l$ such that $(*, \sigma_c^0, l)$ is in $G_x^{\text{MST}}?$
Identity of $l$ such that $(*, \beta_c^0, l)$ is in $G_x^{\text{MST}}?$

Table 1: Guide features for MST<sub>Malt</sub> and Malt<sub>MST</sub>.

by a  $k$ -dimensional feature vector  $\mathbf{f} : X \rightarrow \mathbb{R}^k$ . In the feature-based integration we simply extend the feature vector for one model, called the *base model*, with a certain number of features generated by the other model, which we call the *guide model* in this context. The additional features will be referred to as *guide features*, and the version of the base model trained with the extended feature vector will be called the *guided model*. The idea is that the guided model should be able to learn in which situations to trust the guide features, in order to exploit the complementary strength of the guide model, so that performance can be improved with respect to the base parser. This method of combining classifiers is sometimes referred to as *classifier stacking*.

The exact form of the guide features depend on properties of the base model and will be discussed in sections 3.2–3.3 below, but the overall scheme for the feature-based integration can be described as follows. To train a guided version  $B_C$  of base model  $B$  with guide model  $C$  and training set  $T$ , the guided model is trained, not on the original training set  $T$ , but on a version of  $T$  that has been parsed with the guide model  $C$  under a cross-validation scheme (to avoid overlap with training data for  $C$ ). This means that, for every sentence  $x \in T$ ,  $B_C$  has access at training time to both the gold standard dependency graph  $G_x$  and the graph  $G_x^C$  predicted by  $C$ , and it is the latter that forms the basis for the additional guide features. When parsing a new sentence  $x'$  with  $B_C$ ,  $x'$  is first parsed with model  $C$  (this time trained on the entire training set  $T$ ) to derive  $G_{x'}^C$ , so that the guide features can be extracted also at parsing time.

### 3.2 The Guided Graph-Based Model

The graph-based model, MSTParser, learns a scoring function  $s(i, j, l) \in \mathbb{R}$  over labeled dependencies. More precisely, dependency arcs (or pairs of arcs) are first represented by a high dimensional feature vector  $\mathbf{f}(i, j, l) \in \mathbb{R}^k$ , where  $\mathbf{f}$  is typically a binary feature vector over properties of the arc as well as the surrounding input (McDonald et al., 2005a; McDonald et al., 2006). The score of an arc is defined as a linear classifier  $s(i, j, l) = \mathbf{w} \cdot \mathbf{f}(i, j, l)$ , where  $\mathbf{w}$  is a vector of feature weights to be learned by the model.

For the guided graph-based model, which we call MST<sub>Malt</sub>, this feature representation is modified to include an additional argument  $G_x^{\text{Malt}}$ , which is the dependency graph predicted by MaltParser on the input sentence  $x$ . Thus, the new feature representation will map an arc *and* the entire predicted MaltParser graph to a high dimensional feature representation,  $\mathbf{f}(i, j, l, G_x^{\text{Malt}}) \in \mathbb{R}^{k+m}$ . These  $m$  additional features account for the guide features over the MaltParser output. The specific features used by MST<sub>Malt</sub> are given in table 1. All features are conjoined with the part-of-speech tags of the words involved in the dependency to allow the guided parser to learn weights relative to different surface syntactic environments. Though MSTParser is capable of defining features over pairs of arcs, we restrict the guide features over single arcs as this resulted in higher accuracies during preliminary experiments.

### 3.3 The Guided Transition-Based Model

The transition-based model, MaltParser, learns a scoring function  $s(c, t) \in \mathbb{R}$  over configurations and transitions. The set of training instances for this learning problem is the set of pairs  $(c, t)$  such that  $t$  is the correct transition out of  $c$  in the transition sequence that derives the correct dependency graph  $G_x$  for some sentence  $x$  in the training set  $T$ . Each training instance  $(c, t)$  is represented by a feature vector  $\mathbf{f}(c, t) \in \mathbb{R}^k$ , where features are defined in terms of arbitrary properties of the configuration  $c$ , including the state of the stack  $\sigma_c$ , the input buffer  $\beta_c$ , and the partially built dependency graph  $G_c$ . In particular, many features involve properties of the two target tokens, the token on top of the stack  $\sigma_c$  ( $\sigma_c^0$ ) and the first token in the input buffer  $\beta_c$  ( $\beta_c^0$ ),

which are the two tokens that may become connected by a dependency arc through the transition out of  $c$ . The full set of features used by the base model MaltParser is described in Nivre et al. (2006).

For the guided transition-based model, which we call Malt<sub>MST</sub>, training instances are extended to triples  $(c, t, G_x^{\text{MST}})$ , where  $G_x^{\text{MST}}$  is the dependency graph predicted by the graph-based MSTParser for the sentence  $x$  to which the configuration  $c$  belongs. We define  $m$  additional guide features, based on properties of  $G_x^{\text{MST}}$ , and extend the feature vector accordingly to  $\mathbf{f}(c, t, G_x^{\text{MST}}) \in \mathbb{R}^{k+m}$ . The specific features used by Malt<sub>MST</sub> are given in table 1. Unlike MSTParser, features are not explicitly defined to conjoin guide features with part-of-speech features. These features are implicitly added through the polynomial kernel used to train the SVM.

## 4 Experiments

In this section, we present an experimental evaluation of the two guided models based on data from the CoNLL-X shared task, followed by a comparative error analysis including both the base models and the guided models. The data for the experiments are training and test sets for all thirteen languages from the CoNLL-X shared task on multilingual dependency parsing with training sets ranging in size from 29,000 tokens (Slovene) to 1,249,000 tokens (Czech). The test sets are all standardized to about 5,000 tokens each. For more information on the data sets, see Buchholz and Marsi (2006).

The guided models were trained according to the scheme explained in section 3, with two-fold cross-validation when parsing the training data with the guide parsers. Preliminary experiments suggested that cross-validation with more folds had a negligible impact on the results. Models are evaluated by their *labeled attachment score* (LAS) on the test set, i.e., the percentage of tokens that are assigned both the correct head and the correct label, using the evaluation software from the CoNLL-X shared task with default settings.<sup>4</sup> Statistical significance was assessed using Dan Bikel’s randomized parsing evaluation comparator with the default setting of 10,000 iterations.<sup>5</sup>

<sup>4</sup><http://nextens.uvt.nl/~conll/software.html>

<sup>5</sup><http://www.cis.upenn.edu/~dbikel/software.html>

Language	MST	MST <sub>Malt</sub>	Malt	Malt <sub>MST</sub>
Arabic	66.91	<b>68.64</b> (+1.73)	66.71	67.80 (+1.09)
Bulgarian	87.57	<b>89.05</b> (+1.48)	87.41	88.59 (+1.18)
Chinese	85.90	<b>88.43</b> (+2.53)	86.92	87.44 (+0.52)
Czech	80.18	<b>82.26</b> (+2.08)	78.42	81.18 (+2.76)
Danish	84.79	<b>86.67</b> (+1.88)	84.77	85.43 (+0.66)
Dutch	79.19	<b>81.63</b> (+2.44)	78.59	79.91 (+1.32)
German	87.34	<b>88.46</b> (+1.12)	85.82	87.66 (+1.84)
Japanese	90.71	91.43 (+0.72)	91.65	<b>92.20</b> (+0.55)
Portuguese	86.82	87.50 (+0.68)	87.60	<b>88.64</b> (+1.04)
Slovene	73.44	<b>75.94</b> (+2.50)	70.30	74.24 (+3.94)
Spanish	82.25	<b>83.99</b> (+1.74)	81.29	82.41 (+1.12)
Swedish	82.55	<b>84.66</b> (+2.11)	84.58	84.31 (-0.27)
Turkish	63.19	64.29 (+1.10)	65.58	<b>66.28</b> (+0.70)
Average	80.83	<b>82.53</b> (+1.70)	80.74	82.01 (+1.27)

Table 2: Labeled attachment scores for base parsers and guided parsers (improvement in percentage points).

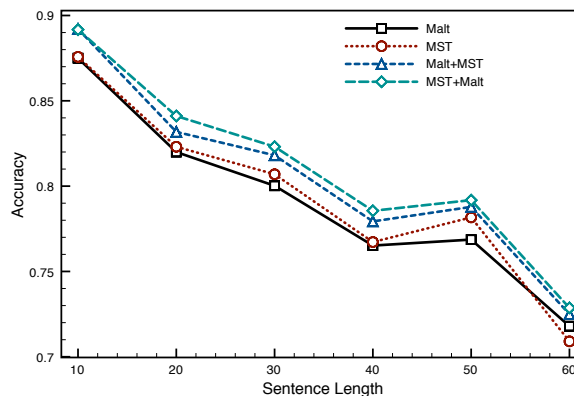


Figure 2: Accuracy relative to sentence length.

### 4.1 Results

Table 2 shows the results, for each language and on average, for the two base models (MST, Malt) and for the two guided models (MST<sub>Malt</sub>, Malt<sub>MST</sub>). First of all, we see that both guided models show a very consistent increase in accuracy compared to their base model, even though the extent of the improvement varies across languages from about half a percentage point (Malt<sub>MST</sub> on Chinese) up to almost four percentage points (Malt<sub>MST</sub> on Slovene).<sup>6</sup> It is thus quite clear that both models have the capacity to learn from features generated by the other model. However, it is also clear that the graph-based MST model shows a somewhat larger improvement, both on average and for all languages except Czech,

<sup>6</sup>The only exception to this pattern is the result for Malt<sub>MST</sub> on Swedish, where we see an unexpected drop in accuracy compared to the base model.

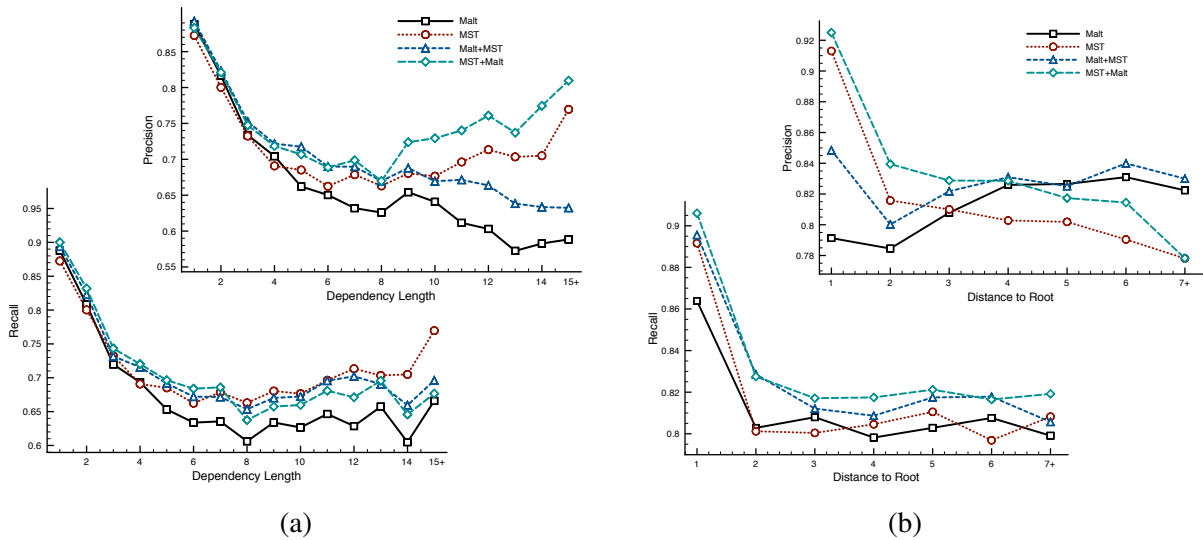


Figure 3: Dependency arc precision/recall relative to predicted/gold for (a) dependency length and (b) distance to root.

German, Portuguese and Slovene. Finally, given that the two base models had the previously best performance for these data sets, the guided models achieve a substantial improvement of the state of the art. While there is no statistically significant difference between the two base models, they are both outperformed by  $\text{Malt}_{\text{MST}}$  ( $p < 0.0001$ ), which in turn has significantly lower accuracy than  $\text{MST}_{\text{Malt}}$  ( $p < 0.0005$ ).

An extension to the models described so far would be to iteratively integrate the two parsers in the spirit of pipeline iteration (Hollingshead and Roark, 2007). For example, one could start with a Malt model, use it to train a guided  $\text{MST}_{\text{Malt}}$  model, then use that as the guide to train a  $\text{Malt}_{\text{MST}_{\text{Malt}}}$  model, etc. We ran such experiments, but found that accuracy did not increase significantly and in some cases decreased slightly. This was true regardless of which parser began the iterative process. In retrospect, this result is not surprising. Since the initial integration effectively incorporates knowledge from both parsing systems, there is little to be gained by adding additional parsers in the chain.

## 4.2 Analysis

The experimental results presented so far show that feature-based integration is a viable approach for improving the accuracy of both graph-based and transition-based models for dependency parsing, but they say very little about how the integration benefits

the two models and what aspects of the parsing process are improved as a result. In order to get a better understanding of these matters, we replicate parts of the error analysis presented by McDonald and Nivre (2007), where parsing errors are related to different structural properties of sentences and their dependency graphs. For each of the four models evaluated, we compute error statistics for labeled attachment over all twelve languages together.

Figure 2 shows accuracy in relation to sentence length, binned into ten-word intervals (1–10, 11–20, etc.). As expected, Malt and MST have very similar accuracy for short sentences but Malt degrades more rapidly with increasing sentence length because of error propagation (McDonald and Nivre, 2007). The guided models,  $\text{Malt}_{\text{MST}}$  and  $\text{MST}_{\text{Malt}}$ , behave in a very similar fashion with respect to each other but both outperform their base parser over the entire range of sentence lengths. However, except for the two extreme data points (0–10 and 51–60) there is also a slight tendency for  $\text{Malt}_{\text{MST}}$  to improve more for longer sentences and for  $\text{MST}_{\text{Malt}}$  to improve more for short sentences, which indicates that the feature-based integration allows one parser to exploit the strength of the other.

Figure 3(a) plots precision (top) and recall (bottom) for dependency arcs of different lengths (predicted arcs for precision, gold standard arcs for recall). With respect to recall, the guided models appear to have a slight advantage over the base mod-

Part of Speech	MST	MST <sub>Malt</sub>	Malt	Malt <sub>MST</sub>
Verb	82.6	<b>85.1</b> (2.5)	81.9	84.3 (2.4)
Noun	80.0	81.7 (1.7)	80.7	<b>81.9</b> (1.2)
Pronoun	88.4	<b>89.4</b> (1.0)	89.2	89.3 (0.1)
Adjective	89.1	<b>89.6</b> (0.5)	87.9	89.0 (1.1)
Adverb	78.3	<b>79.6</b> (1.3)	77.4	78.1 (0.7)
Adposition	69.9	<b>71.5</b> (1.6)	68.8	70.7 (1.9)
Conjunction	73.1	<b>74.9</b> (1.8)	69.8	72.5 (2.7)

Table 3: Accuracy relative to dependent part of speech (improvement in percentage points).

els for short and medium distance arcs. With respect to precision, however, there are two clear patterns. First, the graph-based models have better precision than the transition-based models when predicting long arcs, which is compatible with the results of McDonald and Nivre (2007). Secondly, both the guided models have better precision than their base model and, for the most part, also their guide model. In particular MST<sub>Malt</sub> outperforms MST and is comparable to Malt for short arcs. More interestingly, Malt<sub>MST</sub> outperforms both Malt and MST for arcs up to length 9, which provides evidence that Malt<sub>MST</sub> has learned specifically to trust the guide features from MST for longer dependencies. The reason that accuracy does not improve for dependencies of length greater than 9 is probably that these dependencies are too rare for Malt<sub>MST</sub> to learn from the guide parser in these situations.

Figure 3(b) shows precision (top) and recall (bottom) for dependency arcs at different distances from the root (predicted arcs for precision, gold standard arcs for recall). Again, we find the clearest patterns in the graphs for precision, where Malt has very low precision near the root but improves with increasing depth, while MST shows the opposite trend (McDonald and Nivre, 2007). Considering the guided models, it is clear that Malt<sub>MST</sub> improves in the direction of its guide model, with a 5-point increase in precision for dependents of the root and smaller improvements for longer distances. Similarly, MST<sub>Malt</sub> improves precision in the range where its base parser is inferior to Malt and for distances up to 4 has an accuracy comparable to or higher than its guide parser Malt. This again provides evidence that the guided parsers are learning from their guide models.

Table 3 gives the accuracy for arcs relative to de-

pendent part-of-speech. As expected, we see that MST does better than Malt for all categories except nouns and pronouns (McDonald and Nivre, 2007). But we also see that the guided models in all cases improve over their base parser and, in most cases, also over their guide parser. The general trend is that MST improves more than Malt, except for adjectives and conjunctions, where Malt has a greater disadvantage from the start and therefore benefits more from the guide features.

Considering the results for parts of speech, as well as those for dependency length and root distance, it is interesting to note that the guided models often improve even in situations where their base parsers are more accurate than their guide models. This suggests that the improvement is not a simple function of the raw accuracy of the guide model but depends on the fact that labeled dependency decisions interact in inference algorithms for both graph-based and transition-based parsing systems. Thus, if a parser can improve its accuracy on one class of dependencies, e.g., longer ones, then we can expect to see improvements on all types of dependencies – as we do.

The interaction between different decisions may also be part of the explanation why MST benefits more from the feature-based integration than Malt, with significantly higher accuracy for MST<sub>Malt</sub> than for Malt<sub>MST</sub> as a result. Since inference is global (or practically global) in the graph-based model, an improvement in one type of dependency has a good chance of influencing the accuracy of other dependencies, whereas in the transition-based model, where inference is greedy, some of these additional benefits will be lost because of error propagation. This is reflected in the error analysis in the following recurrent pattern: Where Malt does well, Malt<sub>MST</sub> does only slightly better. But where MST is good, MST<sub>Malt</sub> is often significantly better.

Another part of the explanation may have to do with the learning algorithms used by the systems. Although both Malt and MST use discriminative algorithms, Malt uses a batch learning algorithm (SVM) and MST uses an online learning algorithm (MIRA). If the original rich feature representation of Malt is sufficient to separate the training data, regularization may force the weights of the guided features to be small (since they are not needed at training time). On the other hand, an online learn-



ing algorithm will recognize the guided features as strong indicators early in training and give them a high weight as a result. Features with high weight early in training tend to have the most impact on the final classifier due to both weight regularization and averaging. This is in fact observed when inspecting the weights of  $MST_{\text{Malt}}$ .

## 5 Related Work

Combinations of graph-based and transition-based models for data-driven dependency parsing have previously been explored by Sagae and Lavie (2006), who report improvements of up to 1.7 percentage points over the best single parser when combining three transition-based models and one graph-based model for unlabeled dependency parsing, evaluated on data from the Penn Treebank. The combined parsing model is essentially an instance of the graph-based model, where arc scores are derived from the output of the different component parsers. Unlike the models presented here, integration takes place only at parsing time, not at learning time, and requires at least three different base parsers. The same technique was used by Hall et al. (2007) to combine six transition-based parsers in the best performing system in the CoNLL 2007 shared task.

Feature-based integration in the sense of letting a subset of the features for one model be derived from the output of a different model has been exploited for dependency parsing by McDonald (2006), who trained an instance of  $MSTParser$  using features generated by the parsers of Collins (1999) and Charniak (2000), which improved unlabeled accuracy by 1.7 percentage points, again on data from the Penn Treebank. In addition, feature-based integration has been used by Taskar et al. (2005), who trained a discriminative word alignment model using features derived from the IBM models, and by Florian et al. (2004), who trained classifiers on auxiliary data to guide named entity classifiers.

Feature-based integration also has points in common with co-training, which have been applied to syntactic parsing by Sarkar (2001) and Steedman et al. (2003), among others. The difference, of course, is that standard co-training is a weakly supervised method, where guide features *replace*, rather than *complement*, the gold standard annotation during

training. Feature-based integration is also similar to parse re-ranking (Collins, 2000), where one parser produces a set of candidate parses and a second-stage classifier chooses the most likely one. However, feature-based integration is not explicitly constrained to any parse decisions that the guide model might make and only the single most likely parse is used from the guide model, making it significantly more efficient than re-ranking.

Finally, there are several recent developments in data-driven dependency parsing, which can be seen as targeting the specific weaknesses of graph-based and transition-based models, respectively, though without integrating the two models. Thus, Nakagawa (2007) and Hall (2007) both try to overcome the limited feature scope of graph-based models by adding global features, in the former case using Gibbs sampling to deal with the intractable inference problem, in the latter case using a re-ranking scheme. For transition-based models, the trend is to alleviate error propagation by abandoning greedy, deterministic inference in favor of beam search with globally normalized models for scoring transition sequences, either generative (Titov and Henderson, 2007a; Titov and Henderson, 2007b) or conditional (Duan et al., 2007; Johansson and Nugues, 2007).

## 6 Conclusion

In this paper, we have demonstrated how the two dominant approaches to data-driven dependency parsing, graph-based models and transition-based models, can be integrated by letting one model learn from features generated by the other. Our experimental results show that both models consistently improve their accuracy when given access to features generated by the other model, which leads to a significant advancement of the state of the art in data-driven dependency parsing. Moreover, a comparative error analysis reveals that the improvements are largely predictable from theoretical properties of the two models, in particular the tradeoff between global learning and inference, on the one hand, and rich feature representations, on the other. Directions for future research include a more detailed analysis of the effect of feature-based integration, as well as the exploration of other strategies for integrating different parsing models.

## References

- Giuseppe Attardi. 2006. Experiments with a multilingual non-projective dependency parser. In *Proceedings of CoNLL*, pages 166–170.
- Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164.
- Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.
- Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.
- Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML*, pages 175–182.
- Yuan Ding and Martha Palmer. 2004. Synchronous dependency insertion grammars: A grammar formalism for syntax based statistical MT. In *Proceedings of the Workshop on Recent Advances in Dependency Grammar*, pages 90–97.
- Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic parsing action models for multi-lingual dependency parsing. In *Proceedings of EMNLP-CoNLL*, pages 940–946.
- Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*, pages 340–345.
- Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of NAACL/HLT*.
- Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of EMNLP-CoNLL*.
- Keith Hall. 2007. K-best spanning tree parsing. In *Proceedings of ACL*, pages 392–399.
- Kristy Hollingshead and Brian Roark. 2007. Pipeline iteration. In *Proceedings of ACL*, pages 952–959.
- Richard Johansson and Pierre Nugues. 2007. Incremental dependency parsing using online learning. In *Proceedings of EMNLP-CoNLL*, pages 1134–1138.
- Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*, pages 122–131.
- Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*, pages 122–131.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*, pages 523–530.
- Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL*, pages 216–220.
- Ryan McDonald. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.
- Tetsuji Nakagawa. 2007. Multilingual dependency parsing using global features. In *Proceedings of EMNLP-CoNLL*, pages 952–956.
- Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56.
- Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225.
- Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of EMNLP-CoNLL*, pages 915–932.
- Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*, pages 149–160.
- Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of NAACL: Short Papers*, pages 129–132.
- Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of NAACL*, pages 175–182.
- Rion Snow, Dan Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of NIPS*.
- Mark Steedman, Rebecca Hwa, Miles Osborne, and Anoop Sarkar. 2003. Corrected co-training for statistical parsers. In *Proceedings of ICML*, pages 95–102.
- Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of HLT/EMNLP*, pages 73–80.
- Ivan Titov and James Henderson. 2007a. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proceedings of EMNLP-CoNLL*, pages 947–951.
- Ivan Titov and James Henderson. 2007b. A latent variable model for generative dependency parsing. In *Proceedings of IWPT*, pages 144–155.
- Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.

# Efficient, Feature-based, Conditional Random Field Parsing

Jenny Rose Finkel, Alex Kleeman, Christopher D. Manning

Department of Computer Science

Stanford University

Stanford, CA 94305

jrfinkel@cs.stanford.edu, akleeman@stanford.edu, manning@cs.stanford.edu

## Abstract

Discriminative feature-based methods are widely used in natural language processing, but sentence parsing is still dominated by generative methods. While prior feature-based dynamic programming parsers have restricted training and evaluation to artificially short sentences, we present the first general, feature-rich discriminative parser, based on a conditional random field model, which has been successfully scaled to the full WSJ parsing data. Our efficiency is primarily due to the use of stochastic optimization techniques, as well as parallelization and chart prefiltering. On WSJ15, we attain a state-of-the-art F-score of 90.9%, a 14% relative reduction in error over previous models, while being two orders of magnitude faster. On sentences of length 40, our system achieves an F-score of 89.0%, a 36% relative reduction in error over a generative baseline.

## 1 Introduction

Over the past decade, feature-based discriminative models have become the tool of choice for many natural language processing tasks. Although they take much longer to train than generative models, they typically produce higher performing systems, in large part due to the ability to incorporate arbitrary, potentially overlapping features. However, constituency parsing remains an area dominated by generative methods, due to the computational complexity of the problem. Previous work on discriminative parsing falls under one of three approaches. One approach does discriminative reranking of the

$n$ -best list of a generative parser, still usually depending highly on the generative parser score as a feature (Collins, 2000; Charniak and Johnson, 2005). A second group of papers does parsing by a sequence of independent, discriminative decisions, either greedily or with use of a small beam (Ratnaparkhi, 1997; Henderson, 2004). This paper extends the third thread of work, where joint inference via dynamic programming algorithms is used to train models and to attempt to find the globally best parse. Work in this context has mainly been limited to use of artificially short sentences due to exorbitant training and inference times. One exception is the recent work of Petrov et al. (2007), who discriminatively train a grammar with latent variables and do not restrict themselves to short sentences. However their model, like the discriminative parser of Johnson (2001), makes no use of features, and effectively ignores the largest advantage of discriminative training. It has been shown on other NLP tasks that modeling improvements, such as the switch from generative training to discriminative training, usually provide much smaller performance gains than the gains possible from good feature engineering. For example, in (Lafferty et al., 2001), when switching from a generatively trained hidden Markov model (HMM) to a discriminatively trained, linear chain, conditional random field (CRF) for part-of-speech tagging, their error drops from 5.7% to 5.6%. When they add in only a small set of orthographic features, their CRF error rate drops considerably more to 4.3%, and their out-of-vocabulary error rate drops by more than half. This is further supported by Johnson (2001), who saw no parsing gains when switch-

ing from generative to discriminative training, and by Petrov et al. (2007) who saw only small gains of around 0.7% for their final model when switching training methods.

In this work, we provide just such a framework for training a feature-rich discriminative parser. Unlike previous work, we do not restrict ourselves to short sentences, but we do provide results both for training and testing on sentences of length  $\leq 15$  (WSJ15) and for training and testing on sentences of length  $\leq 40$ , allowing previous WSJ15 results to be put in context with respect to most modern parsing literature. Our model is a conditional random field based model. For a rule application, we allow arbitrary features to be defined over the rule categories, span and split point indices, and the words of the sentence. It is well known that constituent length influences parse probability, but PCFGs cannot easily take this information into account. Another benefit of our feature based model is that it effortlessly allows smoothing over previously unseen rules. While the rule may be novel, it will likely contain features which are not. Practicality comes from three sources. We made use of stochastic optimization methods which allow us to find optimal model parameters with very few passes through the data. We found no difference in parser performance between using stochastic gradient descent (SGD), and the more common, but significantly slower, L-BFGS. We also used limited parallelization, and prefiltering of the chart to avoid scoring rules which cannot tile into complete parses of the sentence. This speed-up does not come with a performance cost; we attain an F-score of 90.9%, a 14% relative reduction in errors over previous work on WSJ15.

## 2 The Model

### 2.1 A Conditional Random Field Context Free Grammar (CRF-CFG)

Our parsing model is based on a conditional random field model, however, unlike previous TreeCRF work, e.g., (Cohn and Blunsom, 2005; Jousse et al., 2006), we do not assume a particular tree structure, and instead find the most likely structure *and* labeling. This is similar to conventional probabilistic context-free grammar (PCFG) parsing, with two exceptions: (a) we maximize *conditional* likelihood

of the parse tree, given the sentence, not *joint* likelihood of the tree and sentence; and (b) probabilities are normalized *globally* instead of *locally* – the graphical models depiction of our trees is undirected.

Formally, we have a CFG  $G$ , which consists of (Manning and Schütze, 1999): (i) a set of terminals  $\{w^k, k = 1, \dots, V\}$ ; (ii) a set of nonterminals  $\{N^k, k = 1, \dots, n\}$ ; (iii) a designated start symbol  $ROOT$ ; and (iv) a set of rules,  $\{\rho = N^i \rightarrow \zeta^j\}$ , where  $\zeta^j$  is a sequence of terminals and nonterminals. A PCFG additionally assigns probabilities to each rule  $\rho$  such that  $\forall i \sum_j P(N^i \rightarrow \zeta^j) = 1$ . Our conditional random field CFG (CRF-CFG) instead defines local clique potentials  $\phi(r|s; \theta)$ , where  $s$  is the sentence, and  $r$  contains a one-level subtree of a tree  $t$ , corresponding to a rule  $\rho$ , along with relevant information about the span of words which it encompasses, and, if applicable, the split position (see Figure 1). These potentials are relative to the sentence, unlike a PCFG where rule scores do not have access to words at the leaves of the tree, or even how many words they dominate. We then define a conditional probability distribution over entire trees, using the standard CRF distribution, shown in (1). There is, however, an important subtlety lurking in how we define the partition function. The partition function  $Z_s$ , which makes the probability of all possible parses sum to unity, is defined over all *structures* as well as all labelings of those structures. We define  $\tau(s)$  to be the set of all possible parse trees for the given sentence licensed by the grammar  $G$ .

$$P(t|s; \theta) = \frac{1}{Z_s} \prod_{r \in t} \phi(r|s; \theta) \quad (1)$$

where

$$Z_s = \sum_{t \in \tau(s)} \prod_{r \in t} \phi(r|s; \theta)$$

The above model is not well-defined over all CFGs. Unary rules of the form  $N^i \rightarrow N^j$  can form cycles, leading to infinite unary chains with infinite mass. However, it is standard in the parsing literature to transform grammars into a restricted class of CFGs so as to permit efficient parsing. Binarization of rules (Earley, 1970) is necessary to obtain cubic parsing time, and closure of unary chains is required for finding total probability mass (rather than just best parses) (Stolcke, 1995). To address this issue, we define our model over a restricted class of

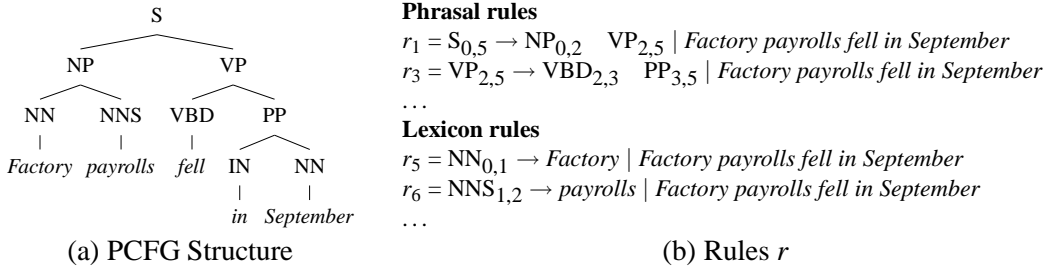


Figure 1: A parse tree and the corresponding rules over which potentials and features are defined.

CFGs which limits unary chains to not have any repeated states. This was done by collapsing all allowed unary chains to single unary rules, and disallowing multiple unary rule applications over the same span.<sup>1</sup> We give the details of our binarization scheme in Section 5. Note that there exists a grammar in this class which is weakly equivalent with any arbitrary CFG.

## 2.2 Computing the Objective Function

Our clique potentials take an exponential form. We have a feature function, represented by  $f(r, s)$ , which returns a vector with the value for each feature. We denote the value of feature  $f_i$  by  $f_i(r, s)$  and our model has a corresponding parameter  $\theta_i$  for each feature. The clique potential function is then:

$$\phi(r|s; \theta) = \exp \sum_i \theta_i f_i(r, s) \quad (2)$$

The log conditional likelihood of the training data  $\mathcal{D}$ , with an additional  $L_2$  regularization term, is then:

$$\mathcal{L}(\mathcal{D}; \theta) = \left( \sum_{(t,s) \in \mathcal{D}} \left( \sum_{r \in \mathcal{R}} \sum_i \theta_i f_i(r, s) \right) - Z_s \right) + \sum_i \frac{\theta_i^2}{2\sigma^2} \quad (3)$$

And the partial derivatives of the log likelihood, with respect to the model weights are, as usual, the difference between the empirical counts and the model expectations:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \left( \sum_{(t,s) \in \mathcal{D}} \left( \sum_{r \in \mathcal{R}} f_i(r, s) \right) - E_{\theta} [f_i|s] \right) + \frac{\theta_i}{\sigma^2} \quad (4)$$

<sup>1</sup>In our implementation of the inside-outside algorithm, we then need to keep two inside and outside scores for each span: one from before and one from after the application of unary rules.

The partition function  $Z_s$  and the partial derivatives can be efficiently computed with the help of the inside-outside algorithm.<sup>2</sup>  $Z_s$  is equal to the inside score of *ROOT* over the span of the entire sentence. To compute the partial derivatives, we walk through each rule, and span/split, and add the outside log-score of the parent, the inside log-score(s) of the child(ren), and the log-score for that rule and span/split.  $Z_s$  is subtracted from this value to get the normalized log probability of that rule in that position. Using the probabilities of each rule application, over each span/split, we can compute the expected feature values (the second term in Equation 4), by multiplying this probability by the value of the feature corresponding to the weight for which we are computing the partial derivative. The process is analogous to the computation of partial derivatives in linear chain CRFs. The complexity of the algorithm for a particular sentence is  $O(n^3)$ , where  $n$  is the length of the sentence.

## 2.3 Parallelization

Unlike (Taskar et al., 2004), our algorithm has the advantage of being easily parallelized (see footnote 7 in their paper). Because the computation of both the log likelihood and the partial derivatives involves summing over each tree individually, the computation can be parallelized by having many clients which each do the computation for one tree, and one central server which aggregates the information to compute the relevant information for a set of trees. Because we use a stochastic optimization method, as discussed in Section 3, we compute the objective for only a small portion of the training data at a time, typically between 15 and 30 sentences. In

<sup>2</sup>In our case the values in the chart are the clique potentials which are non-negative numbers, but not probabilities.

this case the gains from adding additional clients decrease rapidly, because the computation time is dominated by the longest sentences in the batch.

## 2.4 Chart Prefiltering

Training is also sped up by prefiltering the chart. On the inside pass of the algorithm one will see many rules which cannot actually be tiled into complete parses. In standard PCFG parsing it is not worth figuring out which rules are viable at a particular chart position and which are not. In our case however this can make a big difference. We are not just looking up a score for the rule, but must compute all the features, and dot product them with the feature weights, which is far more time consuming. We also have to do an outside pass as well as an inside one, which is sped up by not considering impossible rule applications. Lastly, we iterate through the data multiple times, so if we can compute this information just once, we will save time on all subsequent iterations on that sentence. We do this by doing an inside-outside pass that is just boolean valued to determine which rules are possible at which positions in the chart. We simultaneously compute the features for the possible rules and then save the entire data structure to disk. For all but the shortest of sentences, the disk I/O is easily worth the time compared to re-computation. The first time we see a sentence this method is still about one third faster than if we did not do the prefiltering, and on subsequent iterations the improvement is closer to tenfold.

## 3 Stochastic Optimization Methods

Stochastic optimization methods have proven to be extremely efficient for the training of models involving computationally expensive objective functions like those encountered with our task (Vishwanathan et al., 2006) and, in fact, the on-line backpropagation learning used in the neural network parser of Henderson (2004) is a form of stochastic gradient descent. Standard deterministic optimization routines such as L-BFGS (Liu and Nocedal, 1989) make little progress in the initial iterations, often requiring several passes through the data in order to satisfy sufficient descent conditions placed on line searches. In our experiments SGD converged to a lower objective function value than L-BFGS, however it required far

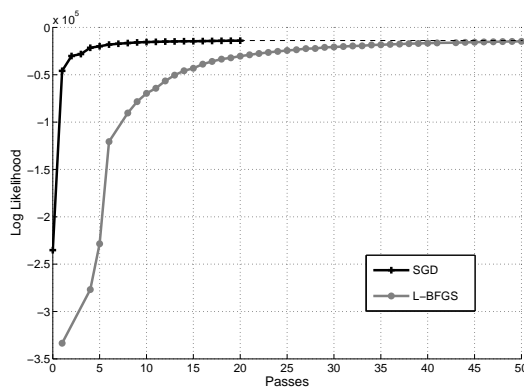


Figure 2: WSJ15 objective value for L-BFGS and SGD versus passes through the data. SGD ultimately converges to a lower objective value, but does equally well on test data.

fewer iterations (see Figure 2) and achieved comparable test set performance to L-BFGS in a fraction of the time. One early experiment on WSJ15 showed a seven time speed up.

### 3.1 Stochastic Function Evaluation

Utilization of stochastic optimization routines requires the implementation of a stochastic objective function. This function,  $\hat{\mathcal{L}}$  is designed to approximate the true function  $\mathcal{L}$  based off a small subset of the training data represented by  $\mathcal{D}_b$ . Here  $b$ , the batch size, means that  $\mathcal{D}_b$  is created by drawing  $b$  training examples, with replacement, from the training set  $\mathcal{D}$ . With this notation we can express the stochastic evaluation of the function as  $\hat{\mathcal{L}}(\mathcal{D}_b; \theta)$ . This stochastic function must be designed to ensure that:

$$\mathbb{E} \left[ \sum_i^n \hat{\mathcal{L}}(\mathcal{D}_b^{(i)}; \theta) \right] = \mathcal{L}(\mathcal{D}; \theta)$$

Note that this property is satisfied, without scaling, for objective functions that sum over the training data, as it is in our case, but any priors must be scaled down by a factor of  $b/|\mathcal{D}|$ . The stochastic gradient,  $\nabla \hat{\mathcal{L}}(\mathcal{D}_b^{(i)}; \theta)$ , is then simply the derivative of the stochastic function value.

### 3.2 Stochastic Gradient Descent

SGD was implemented using the standard update:

$$\theta_{k+1} = \theta_k - \eta_k \nabla \mathcal{L}(\mathcal{D}_b^{(k)}; \theta_k)$$

And employed a gain schedule in the form

$$\eta_k = \eta_0 \frac{\tau}{\tau + k}$$

where parameter  $\tau$  was adjusted such that the gain is halved after five passes through the data. We found that an initial gain of  $\eta_0 = 0.1$  and batch size between 15 and 30 was optimal for this application.

## 4 Features

As discussed in Section 5 we performed experiments on both sentences of length  $\leq 15$  and length  $\leq 40$ . All feature development was done on the length 15 corpus, due to the substantially faster train and test times. This has the unfortunate effect that our features are optimized for shorter sentences and less training data, but we found development on the longer sentences to be infeasible. Our features are divided into two types: *lexicon features*, which are over words and tags, and *grammar features* which are over the local subtrees and corresponding span/split (both have access to the entire sentence). We ran two kinds of experiments: a discriminatively trained model, which used only the rules and no other grammar features, and a feature-based model which did make use of grammar features. Both models had access to the lexicon features. We viewed this as equivalent to the more elaborate, smoothed unknown word models that are common in many PCFG parsers, such as (Klein and Manning, 2003; Petrov et al., 2006).

We preprocessed the words in the sentences to obtain two extra pieces of information. Firstly, each word is annotated with a distributional similarity tag, from a distributional similarity model (Clark, 2000) trained on 100 million words from the British National Corpus and English Gigaword corpus. Secondly, we compute a class for each word based on the unknown word model of Klein and Manning (2003); this model takes into account capitalization, digits, dashes, and other character-level features. The full set of features, along with an explanation of our notation, is listed in Table 1.

## 5 Experiments

For all experiments, we trained and tested on the Penn treebank (PTB) (Marcus et al., 1993). We used

Model	States	Binary Rules	Unary Rules
WSJ15	1,428	5,818	423
WSJ15 relaxed	1,428	22,376	613
WSJ40	7,613	28,240	823

Table 2: Grammar size for each of our models.

the standard splits, training on sections 2 to 21, testing on section 23 and doing development on section 22. Previous work on (non-reranking) discriminative parsing has given results on sentences of length  $\leq 15$ , but most parsing literature gives results on either sentences of length  $\leq 40$ , or all sentences. To properly situate this work with respect to both sets of literature we trained models on both length  $\leq 15$  (WSJ15) and length  $\leq 40$  (WSJ40), and we also tested on all sentences using the WSJ40 models. Our results also provide a context for interpreting previous work which used WSJ15 and not WSJ40.

We used a relatively simple grammar with few additional annotations. Starting with the grammar read off of the training set, we added parent annotations onto each state, including the POS tags, resulting in rules such as  $S-ROOT \rightarrow NP-S VP-S$ . We also added head tag annotations to VPs, in the same manner as (Klein and Manning, 2003). Lastly, for the WSJ40 runs we used a simple, right branching binarization where each active state is annotated with its previous sibling and first child. This is equivalent to children of a state being produced by a second order Markov process. For the WSJ15 runs, each active state was annotated with only its first child, which is equivalent to a first order Markov process. See Table 5 for the number of states and rules produced.

### 5.1 Experiments

For both WSJ15 and WSJ40, we trained a generative model; a discriminative model, which used lexicon features, but no grammar features other than the rules themselves; and a feature-based model which had access to all features. For the length 15 data we also did experiments in which we relaxed the grammar. By this we mean that we added (previously unseen) rules to the grammar, as a means of smoothing. We chose which rules to add by taking existing rules and modifying the parent annotation on the parent of the rule. We used stochastic gradient descent for

Table 1: Lexicon and grammar features.  $w$  is the word and  $t$  the tag.  $r$  represents a particular rule along with span/split information;  $\rho$  is the rule itself,  $r_p$  is the parent of the rule;  $w_b$ ,  $w_s$ , and  $w_e$  are the first, first after the split (for binary rules) and last word that a rule spans in a particular context. All states, including the POS tags, are annotated with parent information;  $b(s)$  represents the base label for a state  $s$  and  $p(s)$  represents the parent annotation on state  $s$ .  $ds(w)$  represents the distributional similarity cluster, and  $lc(w)$  the lower cased version of the word, and  $unk(w)$  the unknown word class.

Lexicon Features	Grammar Features	
$t$		Binary-specific features
$b(t)$	$\rho$	$\langle b(p(r_p)), ds(w_{s-1}, ds w_s) \rangle$
$\langle t, w \rangle$	$\langle b(p(r_p)), ds(w_s) \rangle$	<b>PP feature:</b>
$\langle t, lc(w) \rangle$	$\langle b(p(r_p)), ds(w_e) \rangle$	if right child is a PP then $\langle r, w_s \rangle$
$\langle b(t), w \rangle$	<b>unary?</b>	<b>VP features:</b>
$\langle b(t), lc(w) \rangle$	<b>simplified rule:</b>	if some child is a verb tag, then rule,
$\langle t, ds(w) \rangle$	base labels of states	with that child replaced by the word
$\langle t, ds(w_{-1}) \rangle$	<b>dist sim bigrams:</b>	
$\langle t, ds(w_{+1}) \rangle$	all dist. sim. bigrams below	
$\langle b(t), ds(w) \rangle$	rule, and base parent state	
$\langle b(t), ds(w_{-1}) \rangle$	<b>dist sim bigrams:</b>	Unaries which span one word:
$\langle b(t), ds(w_{+1}) \rangle$	same as above, but trigrams	$\langle r, w \rangle$
$\langle p(t), w \rangle$	<b>heavy feature:</b>	$\langle r, ds(w) \rangle$
$\langle t, unk(w) \rangle$	whether the constituent is “big”	$\langle b(p(r)), w \rangle$
$\langle b(t), unk(w) \rangle$	as described in (Johnson, 2001)	$\langle b(p(r)), ds(w) \rangle$

these experiments; the length 15 models had a batch size of 15 and we allowed twenty passes through the data.<sup>3</sup> The length 40 models had a batch size of 30 and we allowed ten passes through the data. We used development data to decide when the models had converged. Additionally, we provide generative numbers for training on the entire PTB to give a sense of how much performance suffered from the reduced training data (*generative-all* in Table 4).

The full results for WSJ15 are shown in Table 3 and for WSJ40 are shown in Table 4. The WSJ15 models were each trained on a single Dual-Core AMD Opteron™ using three gigabytes of RAM and no parallelization. The discriminatively trained generative model (*discriminative* in Table 3) took approximately 12 minutes per pass through the data, while the feature-based model (*feature-based* in Table 3) took 35 minutes per pass through the data. The feature-based model with the relaxed grammar (*relaxed* in Table 3) took about four times as long as the regular feature-based model. The discrimina-

<sup>3</sup>Technically we did not make passes through the data, because we sampled with replacement to get our batches. By this we mean having seen as many sentences as are in the data, despite having seen some sentences multiple times and some not at all.

tively trained generative WSJ40 model (*discriminative* in Table 4) was trained using two of the same machines, with 16 gigabytes of RAM each for the clients.<sup>4</sup> It took about one day per pass through the data. The feature-based WSJ40 model (*feature-based* in Table 4) was trained using four of these machines, also with 16 gigabytes of RAM each for the clients. It took about three days per pass through the data.

## 5.2 Discussion

The results clearly show that gains came from both the switch from generative to discriminative training, and from the extensive use of features. In Figure 3 we show for an example from section 22 the parse trees produced by our generative model and our feature-based discriminative model, and the correct parse. The parse from the feature-based model better exhibits the right branching tendencies of English. This is likely due to the heavy feature, which encourages long constituents at the end of the sentence. It is difficult for a standard PCFG to learn this aspect of the English language, because the score it assigns to a rule does not take its span into account.

<sup>4</sup>The server does almost no computation.



Model	P	R	F <sub>1</sub>	Exact	Avg CB	0 CB	P	R	F <sub>1</sub>	Exact	Avg CB	0 CB
	development set – length ≤ 15						test set – length ≤ 15					
Taskar 2004	89.7	90.2	90.0	–	–	–	89.1	89.1	89.1	–	–	–
Turian 2007	–	–	–	–	–	–	89.6	89.3	89.4	–	–	–
generative	86.9	85.8	86.4	46.2	0.34	81.2	87.6	85.8	86.7	49.2	0.33	81.9
discriminative	89.1	88.6	88.9	55.5	0.26	85.5	88.9	88.0	88.5	56.6	0.32	85.0
feature-based	90.4	89.3	89.9	59.5	0.24	88.3	91.1	90.2	90.6	61.3	0.24	86.8
relaxed	91.2	90.3	90.7	62.1	0.24	88.1	91.4	90.4	90.9	62.0	0.22	87.9

Table 3: Development and test set results, training and testing on sentences of length  $\leq 15$  from the Penn treebank.

Model	P	R	F <sub>1</sub>	Exact	Avg CB	0 CB	P	R	F <sub>1</sub>	Exact	Avg CB	0 CB
	test set – length ≤ 40						test set – all sentences					
Petrov 2007	–	–	88.8	–	–	–	–	–	88.3	–	–	–
generative	83.5	82.0	82.8	25.5	1.57	53.4	82.8	81.2	82.0	23.8	1.83	50.4
generative-all	83.6	82.1	82.8	25.2	1.56	53.3	–	–	–	–	–	–
discriminative	85.1	84.5	84.8	29.7	1.41	55.8	84.2	83.7	83.9	27.8	1.67	52.8
feature-based	89.2	88.8	89.0	37.3	0.92	65.1	88.2	87.8	88.0	35.1	1.15	62.3

Table 4: Test set results, training on sentences of length  $\leq 40$  from the Penn treebank. The *generative-all* results were trained on all sentences regardless of length

## 6 Comparison With Related Work

The most similar related work is (Johnson, 2001), which did discriminative training of a generative PCFG. The model was quite similar to ours, except that it did not incorporate any features and it required the parameters (which were just scores for rules) to be locally normalized, as with a generatively trained model. Due to training time, they used the ATIS treebank corpus, which is much smaller than even WSJ15, with only 1,088 training sentences, 294 testing sentences, and an average sentence length of around 11. They found no significant difference in performance between their generatively and discriminatively trained parsers. There are two probable reasons for this result. The training set is very small, and it is a known fact that generative models tend to work better for small datasets and discriminative models tend to work better for larger datasets (Ng and Jordan, 2002). Additionally, they made no use of features, one of the primary benefits of discriminative learning.

Taskar et al. (2004) took a large margin approach to discriminative learning, but achieved only small gains. We suspect that this is in part due to the grammar that they chose – the grammar of (Klein and Manning, 2003), which was hand annotated with the intent of optimizing performance of a PCFG. This

grammar is fairly sparse – for any particular state there are, on average, only a few rules with that state as a parent – so the learning algorithm may have suffered because there were few options to discriminate between. Starting with this grammar we found it difficult to achieve gains as well. Additionally, their long training time (several months for WSJ15, according to (Turian and Melamed, 2006)) made feature engineering difficult; they were unable to really explore the space of possible features.

More recent is the work of (Turian and Melamed, 2006; Turian et al., 2007), which improved both the training time and accuracy of (Taskar et al., 2004). They define a simple linear model, use boosted decision trees to select feature conjunctions, and a line search to optimize the parameters. They use an agenda parser, and define their atomic features, from which the decision trees are constructed, over the entire state being considered. While they make extensive use of features, their setup is much more complex than ours and takes substantially longer to train – up to 5 days on WSJ15 – while achieving only small gains over (Taskar et al., 2004).

The most recent similar research is (Petrov et al., 2007). They also do discriminative parsing of length 40 sentences, but with a substantially different setup. Following up on their previous work (Petrov et al., 2006) on grammar splitting, they do discriminative

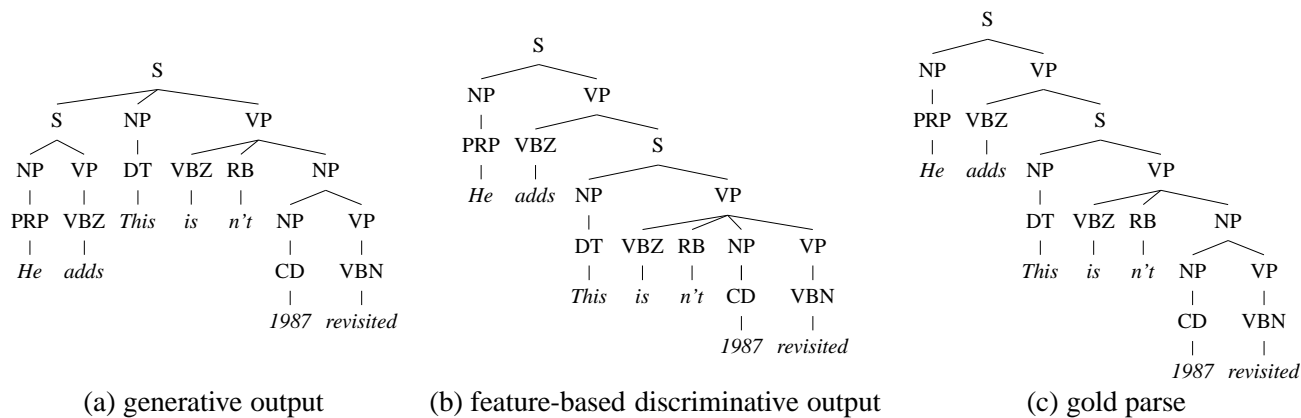


Figure 3: Example output from our generative and feature-based discriminative models, along with the correct parse.

parsing with latent variables, which requires them to optimize a non-convex function. Instead of using a stochastic optimization technique, they use L-BFGS, but do coarse-to-fine pruning to approximate their gradients and log likelihood. Because they were focusing on grammar splitting they, like (Johnson, 2001), did not employ any features, and, like (Taskar et al., 2004), they saw only small gains from switching from generative to discriminative training.

## 7 Conclusions

We have presented a new, feature-rich, dynamic programming based discriminative parser which is simpler, more effective, and faster to train and test than previous work, giving us new state-of-the-art performance when training and testing on sentences of length  $\leq 15$  and the first results for such a parser trained and tested on sentences of length  $\leq 40$ . We also show that the use of SGD for training CRFs performs as well as L-BFGS in a fraction of the time. Other recent work on discriminative parsing has neglected the use of features, despite their being one of the main advantages of discriminative training methods. Looking at how other tasks, such as named entity recognition and part-of-speech tagging, have evolved over time, it is clear that greater gains are to be gotten from developing better features than from better models. We have provided just such a framework for improving parsing performance.

## Acknowledgments

Many thanks to Teg Grenager and Paul Heymann for their advice (and their general awesomeness),

and to our anonymous reviewers for helpful comments.

This paper is based on work funded in part by the Defense Advanced Research Projects Agency through IBM, by the Disruptive Technology Office (DTO) Phase III Program for Advanced Question Answering for Intelligence (AQUAINT) through Broad Agency Announcement (BAA) N61339-06-R-0034, and by a Scottish Enterprise Edinburgh-Stanford Link grant (R37588), as part of the EASIE project.

## References

- Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine  $n$ -best parsing and maxent discriminative reranking. In *ACL 43*, pages 173–180.
- Alexander Clark. 2000. Inducing syntactic categories by context distribution clustering. In *Proc. of Conference on Computational Natural Language Learning*, pages 91–94, Lisbon, Portugal.
- Trevor Cohn and Philip Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *CoNLL 2005*.
- Michael Collins. 2000. Discriminative reranking for natural language parsing. In *ICML 17*, pages 175–182.
- Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 6(8):451–455.
- James Henderson. 2004. Discriminative training of a neural network statistical parser. In *ACL 42*, pages 96–103.
- Mark Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Meeting of the Association for Computational Linguistics*, pages 314–321.
- Florent Jousse, Rémi Gilleron, Isabelle Tellier, and Marc Tommasi. 2006. Conditional Random Fields for XML

- trees. In *ECML Workshop on Mining and Learning in Graphs*.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the Association of Computational Linguistics (ACL)*.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, pages 282–289. Morgan Kaufmann, San Francisco, CA.
- Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528.
- Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.
- 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 Ng and Michael Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems (NIPS)*.
- Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL 44/COLING 21*, pages 433–440.
- Slav Petrov, Adam Pauls, and Dan Klein. 2007. Discriminative log-linear grammars with latent variables. In *NIPS*.
- Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *EMNLP 2*, pages 1–10.
- Andreas Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21:165–202.
- Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher D. Manning. 2004. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Joseph Turian and I. Dan Melamed. 2006. Advances in discriminative parsing. In *ACL 44*, pages 873–880.
- Joseph Turian, Ben Wellington, and I. Dan Melamed. 2007. Scalable discriminative learning for natural language parsing and translation. In *Advances in Neural Information Processing Systems 19*, pages 1409–1416. MIT Press.
- S. V. N. Vishwanathan, Nichol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. 2006. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML 23*, pages 969–976.

# A Deductive Approach to Dependency Parsing\*

**Carlos Gómez-Rodríguez**  
Departamento de Computación  
Universidade da Coruña, Spain  
cgomezr@udc.es

**John Carroll and David Weir**  
Department of Informatics  
University of Sussex, United Kingdom  
{johnca,davidw}@sussex.ac.uk

## Abstract

We define a new formalism, based on Sikkel's parsing schemata for constituency parsers, that can be used to describe, analyze and compare dependency parsing algorithms. This abstraction allows us to establish clear relations between several existing projective dependency parsers and prove their correctness.

## 1 Introduction

Dependency parsing consists of finding the structure of a sentence as expressed by a set of directed links (dependencies) between words. This is an alternative to constituency parsing, which tries to find a division of the sentence into segments (constituents) which are then broken up into smaller constituents. Dependency structures directly show head-modifier and head-complement relationships which form the basis of predicate argument structure, but are not represented explicitly in constituency trees, while providing a representation in which no non-lexical nodes have to be postulated by the parser. In addition to this, some dependency parsers are able to represent non-projective structures, which is an important feature when parsing free word order languages in which discontinuous constituents are common.

The formalism of parsing schemata (Sikkel, 1997) is a useful tool for the study of constituency parsers since it provides formal, high-level descriptions of parsing algorithms that can be used to prove their formal properties (such as correctness), establish relations between them, derive new parsers from existing ones and obtain efficient implementations automatically (Gómez-Rodríguez et al., 2007). The formalism was initially defined for context-free grammars and later applied to other constituency-based formalisms, such as tree-adjointing grammars

(Alonso et al., 1999). However, since parsing schemata are defined as deduction systems over sets of constituency trees, they cannot be used to describe dependency parsers.

In this paper, we define an analogous formalism that can be used to define, analyze and compare dependency parsers. We use this framework to provide uniform, high-level descriptions for a wide range of well-known algorithms described in the literature, and we show how they formally relate to each other and how we can use these relations and the formalism itself to prove their correctness.

### 1.1 Parsing schemata

Parsing schemata (Sikkel, 1997) provide a formal, simple and uniform way to describe, analyze and compare different constituency-based parsers.

The notion of a parsing schema comes from considering parsing as a deduction process which generates intermediate results called *items*. An initial set of items is directly obtained from the input sentence, and the parsing process consists of the application of inference rules (*deduction steps*) which produce new items from existing ones. Each item contains a piece of information about the sentence's structure, and a successful parsing process will produce at least one *final item* containing a full parse tree for the sentence or guaranteeing its existence.

Items in parsing schemata are formally defined as sets of partial parse trees from a set denoted  $Trees(G)$ , which is the set of all the possible partial parse trees that do not violate the constraints imposed by a grammar  $G$ . More formally, an item set  $\mathcal{I}$  is defined by Sikkel as a quotient set associated with an equivalence relation on  $Trees(G)$ .<sup>1</sup>

Valid parses for a string are represented by items containing complete *marked parse trees* for that string. Given a context-free grammar  $G =$

\*Partially supported by Ministerio de Educación y Ciencia and FEDER (TIN2004-07246-C03, HUM2007-66607-C04), Xunta de Galicia (PGIDIT07SIN005206PR, PGIDIT05PXIC-10501PN, PGIDIT05PXIC30501PN, Rede Galega de Proc. da Linguaxe e RI) and Programa de Becas FPU.

<sup>1</sup>While Shieber et al. (1995) also view parsers as deduction systems, Sikkel formally defines items and related concepts, providing the mathematical tools to reason about formal properties of parsers.

$(N, \Sigma, P, S)$ , a *marked parse tree* for a string  $w_1 \dots w_n$  is any tree  $\tau \in \text{Trees}(G)/\text{root}(\tau) = S \wedge \text{yield}(\tau) = w_1 \dots w_n$ <sup>2</sup>. An item containing such a tree for some arbitrary string is called a *final item*. An item containing such a tree for a particular string  $w_1 \dots w_n$  is called a *correct final item* for that string.

For each input string, a parsing schema's deduction steps allow us to infer a set of items, called *valid items* for that string. A parsing schema is said to be *sound* if all valid final items it produces for any arbitrary string are correct for that string. A parsing schema is said to be *complete* if all correct final items are valid. A *correct parsing schema* is one which is both sound and complete. A correct parsing schema can be used to obtain a working implementation of a parser by using deductive engines such as the ones described by Shieber et al. (1995) and Gómez-Rodríguez et al. (2007) to obtain all valid final items.

## 2 Dependency parsing schemata

Although parsing schemata were initially defined for context-free parsers, they can be adapted to different constituency-based grammar formalisms, by finding a suitable definition of  $\text{Trees}(G)$  for each particular formalism and a way to define deduction steps from its rules. However, parsing schemata are not directly applicable to dependency parsing, since their formal framework is based on constituency trees.

In spite of this problem, many of the dependency parsers described in the literature are constructive, in the sense that they proceed by combining smaller structures to form larger ones until they find a complete parse for the input sentence. Therefore, it is possible to define a variant of parsing schemata, where these structures can be defined as items and the strategies used for combining them can be expressed as inference rules. However, in order to define such a formalism we have to tackle some issues specific to dependency parsers:

- Traditional parsing schemata are used to define grammar-based parsers, in which the parsing process is guided by some set of rules which are used to license deduction steps: for example, an Earley *Predictor* step is tied to a particular grammar rule, and can only be executed if such a rule exists. Some dependency parsers are also grammar-

<sup>2</sup> $w_i$  is shorthand for the *marked terminal*  $(w_i, i)$ . These are used by Sikkil (1997) to link terminal symbols to string positions so that an input sentence can be represented as a set of trees which are used as initial items (hypotheses) for the deduction system. Thus, a sentence  $w_1 \dots w_n$  produces a set of hypotheses  $\{\{w_1(\underline{w}_1)\}, \dots, \{w_n(\underline{w}_n)\}\}$ .

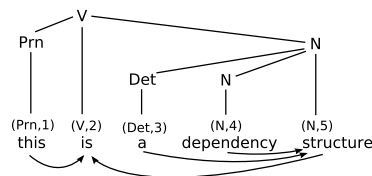


Figure 1: Representation of a dependency structure with a tree. The arrows below the words correspond to its associated dependency graph.

based: for example, those described by Lombardo and Lesmo (1996), Barbero et al. (1998) and Kahane et al. (1998) are tied to the formalizations of dependency grammar using context-free like rules described by Hays (1964) and Gaifman (1965). However, many of the most widely used algorithms (Eisner, 1996; Yamada and Matsumoto, 2003) do not use a formal grammar at all. In these, decisions about which dependencies to create are taken individually, using probabilistic models (Eisner, 1996) or classifiers (Yamada and Matsumoto, 2003). To represent these algorithms as deduction systems, we use the notion of *D-rules* (Covington, 1990). D-rules take the form  $a \rightarrow b$ , which says that word  $b$  can have  $a$  as a dependent. Deduction steps in non-grammar-based parsers can be tied to the D-rules associated with the links they create. In this way, we obtain a representation of the semantics of these parsing strategies that is independent of the particular model used to take the decisions associated with each D-rule.

- The fundamental structures in dependency parsing are *dependency graphs*. Therefore, as items for constituency parsers are defined as sets of partial constituency trees, it is tempting to define items for dependency parsers as sets of partial dependency graphs. However, predictive grammar-based algorithms such as those of Lombardo and Lesmo (1996) and Kahane et al. (1998) have operations which postulate rules and cannot be defined in terms of dependency graphs, since they do not do any modifications to the graph. In order to make the formalism general enough to include these parsers, we define items in terms of sets of partial dependency *trees* as shown in Figure 1. Note that a dependency graph can always be extracted from such a tree.

- Some of the most popular dependency parsing algorithms, like that of Eisner (1996), work by connecting *spans* which can represent *disconnected* dependency graphs. Such spans cannot be represented by a single dependency tree. Therefore, our formalism allows items to be sets of *forests* of partial dependency trees, instead of sets of trees.

Taking these considerations into account, we define the concepts that we need to describe item sets for dependency parsers:

Let  $\Sigma$  be an alphabet of terminal symbols.

**Partial dependency trees:** We define the set of *partial dependency trees* (*D-trees*) as the set of finite trees where children of each node have a left-to-right ordering, each node is labelled with an element of  $\Sigma \cup (\Sigma \times \mathbb{N})$ , and the following conditions hold:

- All nodes labelled with marked terminals  $\underline{w}_i \in (\Sigma \times \mathbb{N})$  are leaves,
- Nodes labelled with terminals  $w \in \Sigma$  do not have more than one daughter labelled with a marked terminal, and if they have such a daughter node, it is labelled  $\underline{w}_i$  for some  $i \in \mathbb{N}$ ,
- Left siblings of nodes labelled with a marked terminal  $\underline{w}_k$  do not have any daughter labelled  $\underline{w}_j$  with  $j \geq k$ . Right siblings of nodes labelled with a marked terminal  $\underline{w}_k$  do not have any daughter labelled  $\underline{w}_j$  with  $j \leq k$ .

We denote the root node of a partial dependency tree  $t$  as  $root(t)$ . If  $root(t)$  has a daughter node labelled with a marked terminal  $\underline{w}_h$ , we will say that  $\underline{w}_h$  is the *head* of the tree  $t$ , denoted by  $head(t)$ . If all nodes labelled with terminals in  $t$  have a daughter labelled with a marked terminal,  $t$  is *grounded*.

**Relationship between trees and graphs:** Let  $t \in D\text{-trees}$  be a partial dependency tree;  $g(t)$ , its associated dependency graph, is a graph  $(V, E)$

- $V = \{\underline{w}_i \in (\Sigma \times \mathbb{N}) \mid \underline{w}_i \text{ is the label of a node in } t\}$ ,
- $E = \{(\underline{w}_i, \underline{w}_j) \in (\Sigma \times \mathbb{N})^2 \mid C, D \text{ are nodes in } t \text{ such that } D \text{ is a daughter of } C, \underline{w}_j \text{ the label of a daughter of } C, \underline{w}_i \text{ the label of a daughter of } D\}$ .

**Projectivity:** A partial dependency tree  $t \in D\text{-trees}$  is *projective* iff  $yield(t)$  cannot be written as  $\dots \underline{w}_i \dots \underline{w}_j \dots$  where  $i \geq j$ .

It is easy to verify that the dependency graph  $g(t)$  is projective with respect to the linear order of marked terminals  $\underline{w}_i$ , according to the usual definition of projectivity found in the literature (Nivre, 2006), if and only if the tree  $t$  is projective.

**Parse tree:** A partial dependency tree  $t \in D\text{-trees}$  is a *parse tree* for a given string  $w_1 \dots w_n$  if its yield is a permutation of  $\underline{w}_1 \dots \underline{w}_n$ . If its yield is exactly  $\underline{w}_1 \dots \underline{w}_n$ , we will say it is a *projective parse tree* for the string.

**Item set:** Let  $\delta \subseteq D\text{-trees}$  be the set of dependency trees which are acceptable according to a given grammar  $G$  (which may be a grammar of D-rules or of CFG-like rules, as explained above). We

define an *item set* for dependency parsing as a set  $\mathcal{I} \subseteq \Pi$ , where  $\Pi$  is a partition of  $2^\delta$ .

Once we have this definition of an item set for dependency parsing, the remaining definitions are analogous to those in Sikkel’s theory of constituency parsing (Sikkel, 1997), so we will not include them here in full detail. A *dependency parsing system* is a deduction system  $(\mathcal{I}, H, D)$  where  $\mathcal{I}$  is a dependency item set as defined above,  $H$  is a set containing *initial items* or *hypotheses*, and  $D \subseteq (2^{H \cup \mathcal{I}} \times \mathcal{I})$  is a set of *deduction steps* defining an inference relation  $\vdash$ .

*Final items* in this formalism will be those containing some forest  $F$  containing a parse tree for some arbitrary string. An item containing such a tree for a particular string  $w_1 \dots w_n$  will be called a *correct final item* for that string in the case of nonprojective parsers. When defining projective parsers, correct final items will be those containing *projective* parse trees for  $w_1 \dots w_n$ . This distinction is relevant because the concepts of soundness and correctness of parsing schemata are based on correct final items (cf. section 1.1), and we expect correct projective parsers to produce only projective structures, while nonprojective parsers should find all possible structures including nonprojective ones.

### 3 Some practical examples

#### 3.1 Col96 (Collins, 96)

One of the most straightforward projective dependency parsing strategies is the one described by Collins (1996), directly based on the CYK parsing algorithm. This parser works with dependency trees which are linked to each other by creating links between their heads. Its item set is defined as  $\mathcal{I}_{Col96} = \{[i, j, h] \mid 1 \leq i \leq h \leq j \leq n\}$ , where an item  $[i, j, h]$  is defined as the set of forests containing a single projective dependency tree  $t$  such that  $t$  is grounded,  $yield(t) = \underline{w}_i \dots \underline{w}_j$  and  $head(t) = \underline{w}_h$ .

For an input string  $w_1 \dots w_n$ , the set of hypotheses is  $H = \{[i, i, i] \mid 0 \leq i \leq n + 1\}$ , i.e., the set of forests containing a single dependency tree of the form  $w_i(\underline{w}_i)$ . This same set of hypotheses can be used for all the parsers, so we will not make it explicit for subsequent schemata.<sup>3</sup>

The set of final items is  $\{[1, n, h] \mid 1 \leq h \leq n\}$ : these items trivially represent parse trees for the input sentence, where  $w_h$  is the sentence’s head. The deduction steps are shown in Figure 2.

<sup>3</sup>Note that the words  $w_0$  and  $w_{n+1}$  used in the definition do not appear in the input: these are dummy terminals that we will call beginning of sentence (BOS) and end of sentence (EOS) marker, respectively; and will be needed by some parsers.

<p><b>Col96 (Collins, 96):</b></p> $R\text{-Link} \frac{\frac{[i, j, h_1]}{[j+1, k, h_2]}}{[i, k, h_2]} w_{h_1} \rightarrow w_{h_2}$ $L\text{-Link} \frac{\frac{[i, j, h_1]}{[j+1, k, h_2]}}{[i, k, h_1]} w_{h_2} \rightarrow w_{h_1}$	<p><b>Eis96 (Eisner, 96):</b></p> $Initter \frac{[i, i, i]}{[i, i+1, F, F]} [i+1, i+1, i+1]$ $R\text{-Link} \frac{[i, j, F, F]}{[i, j, T, F]} w_i \rightarrow w_j$ $L\text{-Link} \frac{[i, j, F, F]}{[i, j, F, T]} w_j \rightarrow w_i$ $CombineSpans \frac{\frac{[i, j, b, c]}{[j, k, not(c), d]}}{[i, k, b, d]}$	<p><b>ES99 (Eisner and Satta, 99):</b></p> $R\text{-Link} \frac{[i, j, i]}{[i, k, k]} \frac{[j+1, k, k]}{[j+1, k, k]} w_i \rightarrow w_k$ $L\text{-Link} \frac{[i, j, i]}{[i, k, i]} \frac{[j+1, k, k]}{[j+1, k, k]} w_k \rightarrow w_i$ $R\text{-Combiner} \frac{[i, j, i]}{[i, k, i]} \frac{[j, k, j]}{[j, k, j]}$ $L\text{-Combiner} \frac{[i, j, j]}{[i, k, k]} \frac{[j, k, k]}{[j, k, k]}$
<p><b>YM03 (Yamada and Matsumoto, 2003):</b></p> $Initter \frac{[i, i, i]}{[i, i+1]} \frac{[i+1, i+1, i+1]}{[i, i+1]}$ $R\text{-Link} \frac{[i, j]}{[j, k]} w_j \rightarrow w_k \quad L\text{-Link} \frac{[i, j]}{[j, k]} w_j \rightarrow w_i$	<p><b>LL96 (Lombardo and Lesmo, 96):</b></p> $Initter \frac{[(S), 1, 0]}{[(S), 1, 0]} *_{(S) \in P} \text{Predictor} \frac{[A(\alpha.B\beta), i, j]}{[B(\gamma), j+1, j]} B(\gamma) \in P$ $Scanner \frac{[A(\alpha \star \beta), i, h-1]}{[A(\alpha \star \beta), i, h]} \frac{[h, h, h]}{[h, h, h]} w_h \text{ IS } A$ $Completer \frac{[A(\alpha.B\beta), i, j]}{[A(\alpha.B\beta), i, k]} \frac{[B(\gamma), j+1, k]}{[B(\gamma), j+1, k]}$	

Figure 2: Deduction steps of the parsing schemata for some well-known dependency parsers.

As we can see, we use D-rules as side conditions for deduction steps, since this parsing strategy is not grammar-based. Conceptually, the schema we have just defined describes a recogniser: given a set of D-rules and an input string  $w_1 \dots w_n$ , the sentence can be parsed (projectively) under those D-rules if and only if this deduction system can infer a correct final item. However, when executing this schema with a deductive engine, we can recover the parse forest by following back pointers in the same way as is done with constituency parsers (Billot and Lang, 1989).

Of course, boolean D-rules are of limited interest in practice. However, this schema provides a formalization of a parsing strategy which is independent of the way linking decisions are taken in a particular implementation. In practice, statistical models can be used to decide whether a step linking words  $a$  and  $b$  (i.e., having  $a \rightarrow b$  as a side condition) is executed or not, and probabilities can be attached to items in order to assign different weights to different analyses of the sentence. The same principle applies to the rest of D-rule-based parsers described in this paper.

### 3.2 Eis96 (Eisner, 96)

By counting the number of free variables used in each deduction step of Collins’ parser, we can conclude that it has a time complexity of  $O(n^5)$ . This complexity arises from the fact that a parentless word (head) may appear in any position in the partial results generated by the parser; the complexity can be reduced to  $O(n^3)$  by ensuring that parentless words can only appear at the first or last position of an item. This is the principle behind the parser defined by Eisner (1996), which is still in wide use today (Corston-Oliver et al., 2006; McDonald et al.,

2005a).

The item set for Eisner’s parsing schema is  $\mathcal{I}_{Eis96} = \{[i, j, T, F] \mid 0 \leq i \leq j \leq n\} \cup \{[i, j, F, T] \mid 0 \leq i \leq j \leq n\} \cup \{[i, j, F, F] \mid 0 \leq i \leq j \leq n\}$ , where each item  $[i, j, T, F]$  is defined as the item  $[i, j, j] \in \mathcal{I}_{Col96}$ , each item  $[i, j, F, T]$  is defined as the item  $[i, j, i] \in \mathcal{I}_{Col96}$ , and each item  $[i, j, F, F]$  is defined as the set of forests of the form  $\{t_1, t_2\}$  such that  $t_1$  and  $t_2$  are grounded,  $head(t_1) = \underline{w}_i$ ,  $head(t_2) = \underline{w}_j$ , and  $\exists k \in \mathbb{N}(i \leq k < j) / yield(t_1) = \underline{w}_i \dots \underline{w}_k \wedge yield(t_2) = \underline{w}_{k+1} \dots \underline{w}_j$ .

Note that the flags  $b, c$  in an item  $[i, j, b, c]$  indicate whether the words in positions  $i$  and  $j$ , respectively, have a parent in the item or not. Items with one of the flags set to  $T$  represent dependency trees where the word in position  $i$  or  $j$  is the head, while items with both flags set to  $F$  represent pairs of trees headed at positions  $i$  and  $j$ , and therefore correspond to disconnected dependency graphs.

Deduction steps<sup>4</sup> are shown in Figure 2. The set of final items is  $\{[0, n, F, T]\}$ . Note that these items represent dependency trees rooted at the BOS marker  $w_0$ , which acts as a “dummy head” for the sentence. In order for the algorithm to parse sentences correctly, we will need to define D-rules to allow  $w_0$  to be linked to the real sentence head.

### 3.3 ES99 (Eisner and Satta, 99)

Eisner and Satta (1999) define an  $O(n^3)$  parser for split head automaton grammars that can be used

<sup>4</sup>Alternatively, we could consider items of the form  $[i, i+1, F, F]$  to be hypotheses for this parsing schema, so we would not need an *Initter* step. However, we have chosen to use a standard set of hypotheses valid for all parsers because this allows for more straightforward proofs of relations between schemata.

for dependency parsing. This algorithm is conceptually simpler than Eis96, since it only uses items representing single dependency trees, avoiding items of the form  $[i, j, F, F]$ . Its item set is  $\mathcal{I}_{ES99} = \{[i, j, i] \mid 0 \leq i \leq j \leq n\} \cup \{[i, j, j] \mid 0 \leq i \leq j \leq n\}$ , where items are defined as in Collins’ parsing schema.

Deduction steps are shown in Figure 2, and the set of final items is  $\{[0, n, 0]\}$ . (Parse trees have  $w_0$  as their head, as in the previous algorithm).

Note that, when described for head automaton grammars as in Eisner and Satta (1999), this algorithm seems more complex to understand and implement than the previous one, as it requires four different kinds of items in order to keep track of the state of the automata used by the grammars. However, this abstract representation of its underlying semantics as a dependency parsing schema shows that this parsing strategy is in fact conceptually simpler for dependency parsing.

### 3.4 YM03 (Yamada and Matsumoto, 2003)

Yamada and Matsumoto (2003) define a deterministic, shift-reduce dependency parser guided by support vector machines, which achieves over 90% dependency accuracy on section 23 of the Penn treebank. Parsing schemata are not suitable for directly describing deterministic parsers, since they work at a high abstraction level where a set of operations are defined without imposing order constraints on them. However, many deterministic parsers can be viewed as particular optimisations of more general, nondeterministic algorithms. In this case, if we represent the actions of the parser as deduction steps while abstracting from the deterministic implementation details, we obtain an interesting nondeterministic parser.

Actions in Yamada and Matsumoto’s parser create links between two target nodes, which act as heads of neighbouring dependency trees. One of the actions creates a link where the left target node becomes a child of the right one, and the head of a tree located directly to the left of the target nodes becomes the new left target node. The other action is symmetric, performing the same operation with a right-to-left link. An  $O(n^3)$  nondeterministic parser generalising this behaviour can be defined by using an item set  $\mathcal{I}_{YM03} = \{[i, j] \mid 0 \leq i \leq j \leq n + 1\}$ , where each item  $[i, j]$  is defined as the item  $[i, j, F, F]$  in  $\mathcal{I}_{Eis96}$ ; and the deduction steps are shown in Figure 2.

The set of final items is  $\{[0, n + 1]\}$ . In order for this set to be well-defined, the grammar must have

no D-rules of the form  $w_i \rightarrow w_{n+1}$ , i.e., it must not allow the EOS marker to govern any words. If this is the case, it is trivial to see that every forest in an item of the form  $[0, n + 1]$  must contain a parse tree rooted at the BOS marker and with yield  $w_0 \dots w_n$ .

As can be seen from the schema, this algorithm requires less bookkeeping than any other of the parsers described here.

### 3.5 LL96 (Lombardo and Lesmo, 96) and other Earley-based parsers

The algorithms in the above examples are based on taking individual decisions about dependency links, represented by D-rules. Other parsers, such as that of Lombardo and Lesmo (1996), use grammars with context-free like rules which encode the preferred order of dependents for each given governor, as defined by Gaifman (1965). For example, a rule of the form  $N(Det * PP)$  is used to allow  $N$  to have  $Det$  as left dependent and  $PP$  as right dependent.

The algorithm by Lombardo and Lesmo (1996) is a version of Earley’s context-free grammar parser (Earley, 1970) using Gaifman’s dependency grammar, and can be written by using an item set  $\mathcal{I}_{LomLes} = \{[A(\alpha.\beta), i, j] \mid A(\alpha\beta) \in P \wedge 1 \leq i \leq j \leq n\}$ , where each item  $[A(\alpha.\beta), i, j]$  represents the set of partial dependency trees rooted at  $A$ , where the direct children of  $A$  are  $\alpha\beta$ , and the subtrees rooted at  $\alpha$  have yield  $w_i \dots w_j$ . The deduction steps for the schema are shown in Figure 2, and the final item set is  $\{[(S.), 1, n]\}$ .

As we can see, the schema for Lombardo and Lesmo’s parser resembles the Earley-style parser in Sikkil (1997), with some changes to adapt it to dependency grammar (for example, the *Scanner* always moves the dot over the head symbol \*).

Analogously, other dependency parsing schemata based on CFG-like rules can be obtained by modifying context-free grammar parsing schemata of Sikkil (1997) in a similar way. The algorithm by Barbero et al. (1998) can be obtained from the left-corner parser, and the one by Courtin and Genthial (1998) is a variant of the head-corner parser.

### 3.6 Pseudo-projectivity

Pseudo-projective parsers can generate non-projective analyses in polynomial time by using a projective parsing strategy and postprocessing the results to establish nonprojective links. For example, the algorithm by Kahane et al. (1998) uses a projective parsing strategy like that of LL96, but using the following initializer step instead of the



*Initter* and *Predictor*:<sup>5</sup>  

$$\text{Initter} \frac{}{[A(\alpha), i, i-1]} A(\alpha) \in P \wedge 1 \leq i \leq n$$

#### 4 Relations between dependency parsers

The framework of parsing schemata can be used to establish relationships between different parsing algorithms and to obtain new algorithms from existing ones, or derive formal properties of a parser (such as soundness or correctness) from the properties of related algorithms.

Sikkel (1994) defines several kinds of relations between schemata, which fall into two categories: *generalisation* relations, which are used to obtain more fine-grained versions of parsers, and *filtering* relations, which can be seen as the reverse of generalisation and are used to reduce the number of items and/or steps needed for parsing. He gives a formal definition of each kind of relation. Informally, a parsing schema can be generalised from another via the following transformations:

- Item refinement: We say that  $P_1 \xrightarrow{ir} P_2$  ( $P_2$  is an item refinement of  $P_1$ ) if there is a mapping between items in both parsers such that single items in  $P_1$  are broken into multiple items in  $P_2$  and individual deductions are preserved.
- Step refinement: We say that  $P_1 \xrightarrow{sr} P_2$  if the item set of  $P_1$  is a subset of that of  $P_2$  and every single deduction step in  $P_1$  can be emulated by a sequence of inferences in  $P_2$ .

On the other hand, a schema can be obtained from another by filtering in the following ways:

- Static/dynamic filtering:  $P_1 \xrightarrow{sf/df} P_2$  if the item set of  $P_2$  is a subset of that of  $P_1$  and  $P_2$  allows a subset of the direct inferences in  $P_1$ <sup>6</sup>.
- Item contraction: The inverse of item refinement.  $P_1 \xrightarrow{ic} P_2$  if  $P_2 \xrightarrow{ir} P_1$ .
- Step contraction: The inverse of step refinement.  $P_1 \xrightarrow{sc} P_2$  if  $P_2 \xrightarrow{sr} P_1$ .

All the parsers described in section 3 can be related via generalisation and filtering, as shown in Figure 3. For space reasons we cannot show formal proofs of all the relations, but we sketch the proofs for some of the more interesting cases:

<sup>5</sup>The initialization step as reported in Kahane’s paper is different from this one, as it directly consumes a nonterminal from the input. However, using this step results in an incomplete algorithm. The problem can be fixed either by using the step shown here instead (bottom-up Earley strategy) or by adding an additional step turning it into a bottom-up Left-Corner parser.

<sup>6</sup>Refer to Sikkel (1994) for the distinction between static and dynamic filtering, which we will not use here.

#### 4.1 YM03 $\xrightarrow{sr}$ Eis96

It is easy to see from the schema definitions that  $\mathcal{I}_{YM03} \subseteq \mathcal{I}_{Eis96}$ . In order to prove the relation between these parsers, we need to verify that every deduction step in YM03 can be emulated by a sequence of inferences in Eis96. In the case of the *Initter* step this is trivial, since the *Initters* of both parsers are equivalent. If we write the *R-Link* step in the notation we have used for Eisner items, we have *R-Link*  $\frac{[i, j, F, F] \quad [j, k, F, F]}{[i, k, F, F]} w_j \rightarrow w_k$

This can be emulated in Eisner’s parser by an *R-Link* step followed by a *CombineSpans* step:

$[j, k, F, F] \vdash [j, k, T, F]$  (by *R-Link*),

$[j, k, T, F], [i, j, F, F] \vdash [i, k, F, F]$  (by *CombineSpans*).

Symmetrically, the *L-Link* step in YM03 can be emulated by an *L-Link* followed by a *CombineSpans* in Eis96.

#### 4.2 ES99 $\xrightarrow{sr}$ Eis96

If we write the *R-Link* step in Eisner and Satta’s parser in the notation for Eisner items, we have *R-Link*  $\frac{[i, j, F, T] \quad [j+1, k, T, F]}{[i, k, T, F]} w_i \rightarrow w_k$

This inference can be emulated in Eisner’s parser as follows:

$\vdash [j, j+1, F, F]$  (by *Initter*),

$[i, j, F, T], [j, j+1, F, F] \vdash [i, j+1, F, F]$  (*CombineSpans*),

$[i, j+1, F, F], [j+1, k, T, F] \vdash [i, k, F, F]$  (*CombineSpans*),

$[i, k, F, F] \vdash [i, k, T, F]$  (by *R-Link*).

The proof corresponding to the *L-Link* step is symmetric. As for the *R-Combiner* and *L-Combiner* steps in ES99, it is easy to see that they are particular cases of the *CombineSpans* step in Eis96, and therefore can be emulated by a single application of *CombineSpans*.

Note that, in practice, the relations in sections 4.1 and 4.2 mean that the ES99 and YM03 parsers are superior to Eis96, since they generate fewer items and need fewer steps to perform the same deductions. These two parsers also have the interesting property that they use disjoint item sets (one uses items representing trees while the other uses items representing pairs of trees); and the union of these disjoint sets is the item set used by Eis96. Also note that the optimisation in YM03 comes from contracting deductions in Eis96 so that linking operations are immediately followed by combining operations; while ES99 does the opposite, forcing combining operations to be followed by linking operations.

#### 4.3 Other relations

If we generalise the linking steps in ES99 so that the head of each item can be in any position, we obtain a

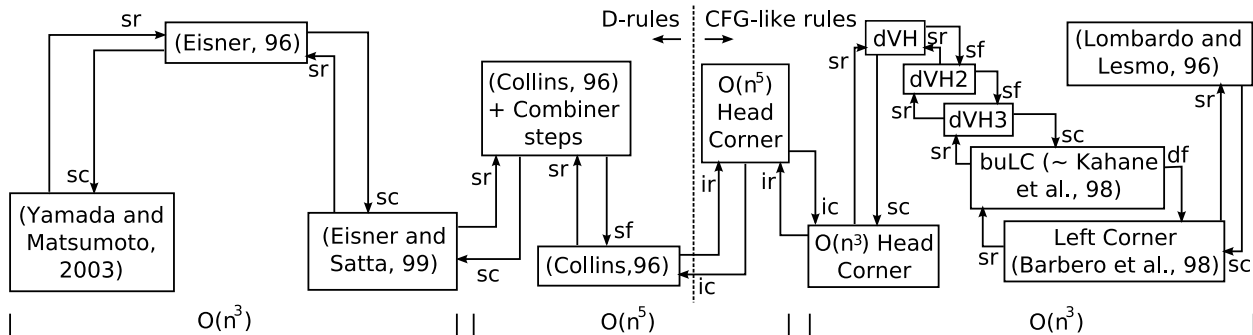


Figure 3: Formal relations between several well-known dependency parsers. Arrows going upwards correspond to generalisation relations, while those going downwards correspond to filtering. The specific subtype of relation is shown in each arrow’s label, following the notation in Section 4.

correct  $O(n^5)$  parser which can be filtered to Col96 just by eliminating the *Combiner* steps.

From Col96, we can obtain an  $O(n^5)$  head-corner parser based on CFG-like rules by an item refinement in which each Collins item  $[i, j, h]$  is split into a set of items  $[A(\alpha.\beta.\gamma), i, j, h]$ . Of course, the formal refinement relation between these parsers only holds if the D-rules used for Collins’ parser correspond to the CFG rules used for the head-corner parser: for every D-rule  $B \rightarrow A$  there must be a corresponding CFG-like rule  $A \rightarrow \dots B \dots$  in the grammar used by the head-corner parser.

Although this parser uses three indices  $i, j, h$ , using CFG-like rules to guide linking decisions makes the  $h$  indices unnecessary, so they can be removed. This simplification is an item contraction which results in an  $O(n^3)$  head-corner parser. From here, we can follow the procedure in Sikkel (1994) to relate this head-corner algorithm to parsers analogous to other algorithms for context-free grammars. In this way, we can refine the head-corner parser to a variant of de Vreught and Honig’s algorithm (Sikkel, 1997), and by successive filters we reach a left-corner parser which is equivalent to the one described by Barbero et al. (1998), and a step contraction of the Earley-based dependency parser LL96. The proofs for these relations are the same as those described in Sikkel (1994), except that the dependency variants of each algorithm are simpler (due to the absence of epsilon rules and the fact that the rules are lexicalised).

## 5 Proving correctness

Another useful feature of the parsing schemata framework is that it provides a formal way to define the correctness of a parser (see last paragraph of Section 1.1) which we can use to prove that our parsers are correct. Furthermore, relations between schemata can be used to derive the correctness of

a schema from that of related ones. In this section, we will show how we can prove that the YM03 and ES99 algorithms are correct, and use that fact to prove the correctness of Eis96.

### 5.1 ES99 is correct

In order to prove the correctness of a parser, we must prove its soundness and completeness (see section 1.1). Soundness is generally trivial to verify, since we only need to check that every individual deduction step in the parser infers a correct consequent item when applied to correct antecedents (i.e., in this case, that steps always generate non-empty items that conform to the definition in 3.3). The difficulty is proving completeness, for which we need to prove that all correct final items are valid (i.e., can be inferred by the schema). To show this, we will prove the stronger result that all correct items are valid.

We will show this by strong induction on the *length* of items, where the length of an item  $\iota = [i, k, h]$  is defined as  $length(\iota) = k - i + 1$ . Correct items of length 1 are the hypotheses of the schema (of the form  $[i, i, i]$ ) which are trivially valid. We will prove that, if all correct items of length  $m$  are valid for all  $1 \leq m < l$ , then items of length  $l$  are also valid.

Let  $[i, k, i]$  be an item of length  $l$  in  $\mathcal{I}_{ES99}$  (thus,  $l = k - i + 1$ ). If this item is correct, then it contains a grounded dependency tree  $t$  such that  $yield(t) = w_i \dots w_k$  and  $head(t) = w_i$ .

By construction, the root of  $t$  is labelled  $w_i$ . Let  $w_j$  be the rightmost daughter of  $w_i$  in  $t$ . Since  $t$  is projective, we know that the yield of  $w_j$  must be of the form  $w_l \dots w_k$ , where  $i < l \leq j \leq k$ . If  $l < j$ , then  $w_l$  is the leftmost transitive dependent of  $w_j$  in  $t$ , and if  $k > j$ , then we know that  $w_k$  is the rightmost transitive dependent of  $w_j$  in  $t$ .

Let  $t_j$  be the subtree of  $t$  rooted at  $w_j$ . Let  $t_1$  be the tree obtained from removing  $t_j$  from  $t$ . Let  $t_2$  be

the tree obtained by removing all the children to the right of  $w_j$  from  $t_j$ , and  $t_3$  be the tree obtained by removing all the children to the left of  $w_j$  from  $t_j$ . By construction,  $t_1$  belongs to a correct item  $[i, l - 1, i]$ ,  $t_2$  belongs to a correct item  $[l, j, j]$  and  $t_3$  belongs to a correct item  $[j, k, j]$ . Since these three items have a length strictly less than  $l$ , by the inductive hypothesis, they are valid. This allows us to prove that the item  $[i, k, i]$  is also valid, since it can be obtained from these valid items by the following inferences:

$[i, l - 1, i], [l, j, j] \vdash [i, j, i]$  (by the *L-Link* step),

$[i, j, i], [j, k, j] \vdash [i, k, i]$  (by the *L-Combiner* step).

This proves that all correct items of length  $l$  which are of the form  $[i, k, i]$  are correct under the inductive hypothesis. The same can be proved for items of the form  $[i, k, k]$  by symmetric reasoning, thus proving that the ES99 parsing schema is correct.

## 5.2 YM03 is correct

In order to prove correctness of this parser, we follow the same procedure as above. Soundness is again trivial to verify. To prove completeness, we use strong induction on the length of items, where the length of an item  $[i, j]$  is defined as  $j - i + 1$ .

The induction step is proven by considering any correct item  $[i, k]$  of length  $l > 2$  ( $l = 2$  is the base case here since items of length 2 are generated by the *Initter* step) and proving that it can be inferred from valid antecedents of length less than  $l$ , so it is valid. To show this, we note that, if  $l > 2$ , either  $w_i$  has at least a right dependent or  $w_k$  has at least a left dependent in the item. Supposing that  $w_i$  has a right dependent, if  $t_1$  and  $t_2$  are the trees rooted at  $w_i$  and  $w_k$  in a forest in  $[i, k]$ , we call  $w_j$  the rightmost daughter of  $w_i$  and consider the following trees:

$v$  = the subtree of  $t_1$  rooted at  $w_j$ ,  $u_1$  = the tree obtained by removing  $v$  from  $t_1$ ,  $u_2$  = the tree obtained by removing all children to the right of  $w_j$  from  $v$ ,  $u_3$  = the tree obtained by removing all children to the left of  $w_j$  from  $v$ .

We observe that the forest  $\{u_1, u_2\}$  belongs to the correct item  $[i, j]$ , while  $\{u_3, t_2\}$  belongs to the correct item  $[j, k]$ . From these two items, we can obtain  $[i, k]$  by using the *L-Link* step. Symmetric reasoning can be applied if  $w_i$  has no right dependents but  $w_k$  has at least a left dependent, and analogously to the case of the previous parser, we conclude that the YM03 parsing schema is correct.

## 5.3 Eis96 is correct

By using the previous proofs and the relationships between schemata that we explained earlier, it is easy to prove that Eis96 is correct: soundness is,

as always, straightforward, and completeness can be proven by using the properties of other algorithms. Since the set of final items in Eis96 and ES99 are the same, and the former is a step refinement of the latter, the completeness of ES99 directly implies the completeness of Eis96.

Alternatively, we can use YM03 to prove the correctness of Eis96 if we redefine the set of final items in the latter to be of the form  $[0, n + 1, F, F]$ , which are equally valid as final items since they always contain parse trees. This idea can be applied to transfer proofs of completeness across any refinement relation.

## 6 Conclusions

We have defined a variant of Sikkel's parsing schemata formalism which allows us to represent dependency parsing algorithms in a simple, declarative way<sup>7</sup>. We have clarified relations between parsers which were originally described very differently. For example, while Eisner presented his algorithm as a dynamic programming algorithm which combines spans into larger spans, Yamada and Matsumoto's works by sequentially executing parsing actions that move a focus point in the input one position to the left or right, (possibly) creating a dependency link. However, in the parsing schemata for these algorithms we can see (and formally prove) that they are related: one is a refinement of the other.

Parsing schemata are also a formal tool that can be used to prove the correctness of parsing algorithms. The relationships between dependency parsers can be exploited to derive properties of a parser from those of others, as we have seen in several examples.

Although the examples in this paper are centered in projective dependency parsing, the formalism does not require projectivity and can be used to represent nonprojective algorithms as well<sup>8</sup>. An interesting line for future work is to use relationships between schemata to find nonprojective parsers that can be derived from existing projective counterparts.

<sup>7</sup>An alternative framework that formally describes some dependency parsers is that of transition systems (McDonald and Nivre, 2007). This model is based on parser configurations and transitions, and has no clear relationship with the approach described here.

<sup>8</sup>Note that spanning tree parsing algorithms based on edge-factored models, such as the one by McDonald et al. (2005b) are not constructive in the sense outlined in Section 2, so the approach described here does not directly apply to them. However, other nonprojective parsers such as (Attardi, 2006) follow a constructive approach and can be analysed deductively.

## References

- Miguel A. Alonso, Eric de la Clergerie, David Cabrero, and Manuel Vilares. 1999. Tabular algorithms for TAG parsing. In *Proc. of the Ninth Conference on European chapter of the Association for Computational Linguistics*, pages 150–157, Bergen, Norway. ACL.
- Giuseppe Attardi. 2006. Experiments with a Multilanguage Non-Projective Dependency Parser. In *Proc. of the Tenth Conference on Natural Language Learning (CoNLL-X)*, pages 166–170, New York, USA. ACL.
- Cristina Barbero, Leonardo Lesmo, Vincenzo Lombardo, and Paola Merlo. 1998. Integration of syntactic and lexical information in a hierarchical dependency grammar. In *Proc. of the Workshop on Dependency Grammars*, pages 58–67, ACL-COLING, Montreal, Canada.
- Sylvie Billot and Bernard Lang. 1989. The structure of shared forest in ambiguous parsing. In *Proc. of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 143–151, Vancouver, British Columbia, Canada, June. ACL.
- Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proc. of the 34th annual meeting on Association for Computational Linguistics*, pages 184–191, Morristown, NJ, USA. ACL.
- Simon Corston-Oliver, Anthony Aue, Kevin Duh, and Eric Ringger. 2006. Multilingual dependency parsing using Bayes Point Machines. In *Proc. of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 160–167, Morristown, NJ, USA. ACL.
- Jacques Courtin and Damien Genthial. 1998. Parsing with dependency relations and robust parsing. In *Proc. of the Workshop on Dependency Grammars*, pages 88–94, ACL-COLING, Montreal, Canada.
- Michael A. Covington. 1990. A dependency parser for variable-word-order languages. Technical Report AI-1990-01, Athens, GA.
- Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.
- Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 457–464, Morristown, NJ, USA. ACL.
- Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, August.
- Haim Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.
- Carlos Gómez-Rodríguez, Jesús Vilares, and Miguel A. Alonso. 2007. Compiling declarative specifications of parsing algorithms. In *Database and Expert Systems Applications*, volume 4653 of *Lecture Notes in Computer Science*, pages 529–538, Springer-Verlag.
- David Hays. 1964. Dependency theory: a formalism and some observations. *Language*, 40:511–525.
- Sylvain Kahane, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *COLING-ACL*, pages 646–652.
- Vincenzo Lombardo and Leonardo Lesmo. 1996. An Earley-type recognizer for dependency grammar. In *Proc. of the 16th conference on Computational linguistics*, pages 723–728, Morristown, NJ, USA. ACL.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *ACL '05: Proc. of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98, Morristown, NJ, USA. ACL.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *HLT '05: Proc. of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. ACL.
- Ryan McDonald and Joakim Nivre. 2007. Characterizing the Errors of Data-Driven Dependency Parsing Models. In *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.
- Joakim Nivre. 2006. *Inductive Dependency Parsing (Text, Speech and Language Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- Stuart M. Shieber, Yves Schabes, and Fernando C.N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.
- Klaas Sikkel. 1994. How to compare the structure of parsing algorithms. In G. Pighizzini and P. San Pietro, editors, *Proc. of ASMICS Workshop on Parsing Theory. Milano, Italy, Oct 1994*, pages 21–39.
- Klaas Sikkel. 1997. *Parsing Schemata — A Framework for Specification and Analysis of Parsing Algorithms*. Texts in Theoretical Computer Science — An EATCS Series. Springer-Verlag, Berlin/Heidelberg/New York.
- Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of 8th International Workshop on Parsing Technologies*, pages 195–206.

# Evaluating a Crosslinguistic Grammar Resource: A Case Study of Wambaya

**Emily M. Bender**

University of Washington

Department of Linguistics

Box 354340

Seattle WA 98195-4340

ebender@u.washington.edu

## Abstract

This paper evaluates the LinGO Grammar Matrix, a cross-linguistic resource for the development of precision broad coverage grammars, by applying it to the Australian language Wambaya. Despite large typological differences between Wambaya and the languages on which the development of the resource was based, the Grammar Matrix is found to provide a significant jump-start in the creation of the grammar for Wambaya: With less than 5.5 person-weeks of development, the Wambaya grammar was able to assign correct semantic representations to 76% of the sentences in a naturally occurring text. While the work on Wambaya identified some areas of refinement for the Grammar Matrix, 59% of the Matrix-provided types were invoked in the final Wambaya grammar, and only 4% of the Matrix-provided types required modification.

## 1 Introduction

Hand-built grammars are often dismissed as too expensive to build on the one hand, and too brittle on the other. Nevertheless, they are key to various NLP applications, including those benefiting from deep natural language understanding (e.g., textual inference (Bobrow et al., 2007)), generation of well-formed output (e.g., natural language weather alert systems (Lareau and Wanner, 2007)) or both (as in machine translation (Oepen et al., 2007)). Of particular interest here are applications concerning endangered languages: Endangered languages represent a case of minimal linguistic resources, typically lacking even moderately-sized corpora, let alone

treebanks. In the best case, one finds well-crafted descriptive grammars, bilingual dictionaries, and a handful of translated texts. The methods of precision grammar engineering are well-suited to taking advantage of such resources. At the same time, the applications of interest in the context of endangered languages emphasize linguistic precision: implemented grammars can be used to enrich existing linguistic documentation, to build grammar checkers in the context of language standardization, and to create software language tutors in the context of language preservation efforts.

The LinGO Grammar Matrix (Bender et al., 2002; Bender and Flickinger, 2005; Drellishak and Bender, 2005) is a toolkit for reducing the cost of creating broad-coverage precision grammars by prepackaging both a cross-linguistic core grammar and a series of libraries of analyses of cross-linguistically variable phenomena, such as major-constituent word order or question formation. The Grammar Matrix was developed initially on the basis of broad-coverage grammars for English (Flickinger, 2000) and Japanese (Siegel and Bender, 2002), and has since been extended and refined as it has been used in the development of broad-coverage grammars for Norwegian (Hellan and Haugereid, 2003), Modern Greek (Kordoni and Neu, 2005), and Spanish (Marimon et al., 2007), as well as being applied to 42 other languages from a variety of language families in a classroom context (Bender, 2007).

This paper aims to evaluate both the utility of the Grammar Matrix in jump-starting precision grammar development and the current state of its cross-linguistic hypotheses through a case study of a

language typologically very different from any of the languages above: the non-Pama-Nyungan Australian language Wambaya (Nordlinger, 1998).

The remainder of this paper is structured as follows: §2 provides background on the Grammar Matrix and Wambaya, and situates the project with respect to related work. §3 presents the implemented grammar of Wambaya, describes its development, and evaluates it against unseen, naturally occurring text. §4 uses the Wambaya grammar and its development as one point of reference to measure the usefulness and cross-linguistic validity of the Grammar Matrix. §5 provides further discussion.

## 2 Background

### 2.1 The LinGO Grammar Matrix

The LinGO Grammar Matrix is situated theoretically within Head-Driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994), a lexicalist, constraint-based framework. Grammars in HPSG are expressed as a collection of typed feature structures which are arranged into a hierarchy such that information shared across multiple lexical entries or construction types is represented only on a single supertype. The Matrix is written in the TDL (type description language) formalism, which is interpreted by the LKB parser, generator, and grammar development environment (Copestake, 2002). It is compatible with the broader range of DELPH-IN tools, e.g., for machine translation (Lønning and Oepen, 2006), treebanking (Oepen et al., 2004) and parse selection (Toutanova et al., 2005).

The Grammar Matrix consists of a cross-linguistic core type hierarchy and a collection of phenomenon-specific libraries. The core type hierarchy defines the basic feature geometry, the ways that heads combine with arguments and adjuncts, linking types for relating syntactic to semantic arguments, and the constraints required to compositionally build up semantic representations in the format of Minimal Recursion Semantics (Copestake et al., 2005; Flickinger and Bender, 2003). The libraries provide collections of analyses for cross-linguistically variable phenomena. The current libraries include analyses of major constituent word order (SOV, SVO, etc), sentential negation, coordination, and yes-no question formation. The Matrix is accessed through

a web-based configuration system<sup>1</sup> which elicits typological information from the user-linguist through a questionnaire and then outputs a grammar consisting of the Matrix core plus selected types and constraints from the libraries according to the specifications in the questionnaire.

### 2.2 Wambaya

Wambaya is a recently extinct language of the West Barkly family from the Northern Territory in Australia (Nordlinger, 1998). Wambaya was selected for this project because of its typological properties and because it is extraordinarily well-documented by Nordlinger in her 1998 descriptive grammar.

Perhaps the most striking feature of Wambaya is its word order: it is a radically non-configurational language with a second position auxiliary/clitic cluster. That is, aside from the constraint that verbal clauses require a clitic cluster (marking subject and object agreement and tense, aspect and mood) in second position, the word order is otherwise free, to the point that noun phrases can be non-contiguous, with head nouns and their modifiers separated by unrelated words. Furthermore, head nouns are generally not required: argument positions can be instantiated by modifiers only, or, if the referent is clear from the context, by no nominal constituent of any kind. It has a rich system of case marking, and adnominal modifiers agree with the heads they modify in case, number, and four genders. An example is given in (1) (Nordlinger, 1998, 223).<sup>2</sup>

- (1) Ngaragana-nguja ngiy-a  
grog-PROP.IV.ACC 3.SG.NM.A-PST  
gujunganjanga-ni jiyawu ngabulu.  
mother.II.ERG give milk.IV.ACC  
'(His) mother gave (him) milk with grog in it.' [wmb]

In (1), *ngaragana-nguja* ('grog-proprietary', or 'having grog') is a modifier of *ngabulu* milk. They agree in case (accusative) and gender (class IV), but they are not contiguous within the sentence.

To relate such discontinuous noun phrases to appropriate semantic representations where 'having-

<sup>1</sup><http://www.delph-in.net/matrix/customize/matrix.cgi>

<sup>2</sup>In this example, the glosses II, IV, and NM indicate gender and ACC and ERG indicate case. A stands for 'agent', PST for 'past', and PROP for 'proprietary'.

grog' and 'milk' are predicated of the same entity requires a departure from the ordinary way that heads are combined with arguments and modifiers combined with heads in HPSG in general and in the Matrix in particular.<sup>3</sup> In the Grammar Matrix, as in most work in HPSG, lexical heads record the dependents they require in valence lists (SUBJ, COMPS, SPR). When a head combines with one of its arguments, the result is a phrase with the same valence requirements as the head daughter, minus the one corresponding to the argument that was just satisfied. In contrast, the project described here has explored a non-cancellation analysis for Wambaya: even after a head combines with one of its arguments, that argument remains on the appropriate valence list of the mother, so that it is visible for further combination with modifiers. In addition, heads can combine directly with modifiers of their arguments (as opposed to just modifiers of themselves).

Argument realization and the combination of heads and modifiers are fairly fundamental aspects of the system implemented in the Matrix. In light of the departure described above, it is interesting to see to what extent the Matrix can still support rapid development of a precision grammar for Wambaya.

### 2.3 Related Work

There are currently many multilingual grammar engineering projects under active development, including ParGram, (Butt et al., 2002; King et al., 2005), the MetaGrammar project (Kinyon et al., 2006), KPML (Bateman et al., 2005), Grammix (Müller, 2007) and OpenCCG (Baldrige et al., 2007). Among approaches to multilingual grammar engineering, the Grammar Matrix's distinguishing characteristics include the deployment of a shared core grammar for crosslinguistically consistent constraints and a series of libraries modeling varying linguistic properties. Thus while other work has successfully exploited grammar porting between typologically related languages (e.g., Kim et al., 2003), to my knowledge, no other grammar porting project has covered the same typological dis-

<sup>3</sup>A linearization-based analysis as suggested by Donohue and Sag (1999) for discontinuous constituents in Warlpiri (another Australian language), is not available, because it relies on disassociating the constituent structure from the surface order of words in a way that is not compatible with the TDL formalism.

tance attempted here. The current project is also situated within a broader trend of using computational linguistics in the service of endangered language documentation (e.g., Robinson et al., 2007, see also [www.emeld.org](http://www.emeld.org)).

## 3 Wambaya grammar

### 3.1 Development

The Wambaya grammar was developed on the basis of the grammatical description in Nordlinger 1998, including the Wambaya-English translation lexicon and glosses of individual example sentences. The development test suite consisted of all 794 distinct positive examples from Ch. 3–8 of the descriptive grammar. This includes elicited examples as well as (sometimes simplified) naturally occurring examples. They range in length from one to thirteen words (mean: 3.65). The test suite was extracted from the descriptive grammar at the beginning of the project and used throughout with only minor refinements as errors in formatting were discovered. The regression testing facilities of [incr tsdb()] allowed for rapid experimentation with alternative analyses as new phenomena were brought into the grammar (cf. Oepen et al., 2002).

With no prior knowledge of this language beyond its most general typological properties, we were able to develop in under 5.5 person-weeks of development time (210 hours) a grammar able to assign appropriate analyses to 91% of the examples in the development set.<sup>4</sup> The 210 hours include 25 hours of an RA's time entering lexical entries, 7 hours spent preparing the development test suite, and 15 hours treebanking (using the LinGO Redwoods software (Oepen et al., 2004) to annotate the intended parse for each item). The remainder of the time was ordinary grammar development work.<sup>5</sup>

In addition, this grammar has relatively low ambiguity, assigning on average 11.89 parses per item in the development set. This reflects the fact that the grammar is modeling grammaticality: the rules are

<sup>4</sup>An additional 6% received some analysis, but not one that matched the translation given in the reference grammar.

<sup>5</sup>These numbers do not include the time put into the original field work and descriptive grammar work. Nordlinger (p.c.) estimates that as roughly 28 linguist-months, plus the native speaker consultants' time.

meant to exclude ungrammatical strings as well as are unwarranted analyses of grammatical strings.

### 3.2 Scope

The grammar encodes mutually interoperable analyses of a wide variety of linguistic phenomena, including:

- Word order: second position clitic cluster, otherwise free word order, discontinuous noun phrases
- Argument optionality: argument positions with no overt head
- Linking of syntactic to semantic arguments
- Case: case assignment by verbs to dependents
- Agreement: subject and object agreement in person and number (and to some extent gender) marked in the clitic cluster, agreement between nouns and adnominal modifiers in case, number and gender
- Lexical adverbs, including manner, time, and location, and adverbs of negation, which vary by clause type (declarative, imperative, or interrogative)
- Derived event modifiers: nominals (nouns, adjectives, noun phrases) used as event modifiers with meaning dependent on their case marking
- Lexical adjectives, including demonstratives adverbs, numerals, and possessive adjectives, as well as ordinary intersective adjectives
- Derived nominal modifiers: modifiers of nouns derived from nouns, adjectives and verbs, including the proprietary, privative, and ‘origin’ constructions
- Subordinate clauses: clausal complements of verbs like “tell” and “remember”, non-finite subordinate clauses such as purposives (“in order to”) and clauses expressing prior or simultaneous events
- Verbless clauses: nouns, adjectives, and adverbs, lexical or derived, functioning as predicates
- Illocutionary force: imperatives, declaratives, and interrogatives (including *wh* questions)
- Coordination: of clauses and noun phrases
- Other: inalienable possession, secondary predicates, causatives of verbs and adjectives

### 3.3 Sample Analysis

This section provides a brief description of the analysis of radical non-configurationality in order to give a sense of the linguistic detail encoded in the Wambaya grammar and give context for the evaluation of the Wambaya grammar and the Grammar Matrix in later sections.

The linguistic analyses encoded in the grammar serve to map the surface strings to semantic representations (in Minimal Recursion Semantics (MRS) format (Copestake et al., 2005)). The MRS in Figure 1 is assigned to the example in (1).<sup>6</sup> It includes the basic propositional structure: a situation of ‘giving’ in which the first argument, or agent, is ‘mother’, the second (recipient) is some third-person entity, and the third (patient), is ‘milk’ which is also related to ‘grog’ through the proprietary relation. It is marked as past tense, and as potentially a statement or a question, depending on the intonation.<sup>7,8</sup>

A simple tree display of the parse giving rise to this MRS is given in Figure 2. The non-branching nodes at the bottom of the tree represent the lexical rules which associate morphosyntactic information with a word according to its suffixes. The general left-branching structure of the tree is a result of the analysis of the second-position clitic cluster: The clitic clusters are treated as argument-composition auxiliaries, which combine with a lexical verb and ‘inherit’ all of the verb’s arguments. The auxiliaries first pick up all dependents to the right, and then combine with exactly one constituent to the left.

The grammar is able to connect *x7* (the index of ‘milk’) to both the ARG3 position of the ‘give’ relation and the ARG1 position of the proprietary relation, despite the separation between *ngaraganaguja* (‘grog-PROP.IV.ACC’) and *ngabulu* (‘milk.IV.ACC’) in the surface structure, as follows: The auxiliary *ngiya* is subject to the constraints in (2), meaning that it combines with a verb as its first complement and then the verb’s complements as its remaining complements.<sup>9</sup> The auxiliary can combine with its complements in any order, thanks to a series of head-complement rules which realize the *n*th element of

<sup>6</sup>The grammar in fact finds 42 parses for this example. The one associated with the MRS in Figure 1 best matches the intended interpretation as indicated by the gloss of the example.

<sup>7</sup>The relations are given English predicate names for the convenience of the grammar developer, and these are not intended as any kind of interlingua.

<sup>8</sup>This MRS is ‘fragmented’ in the sense that the labels of several of the elementary predications (eps) are not related to any argument position of any other ep. This is related to the fact that the grammar doesn’t yet introduce quantifiers for any of the nominal arguments.

<sup>9</sup>In this and other attribute value matrices displayed, feature paths are abbreviated and detail not relevant to the current point is suppressed.



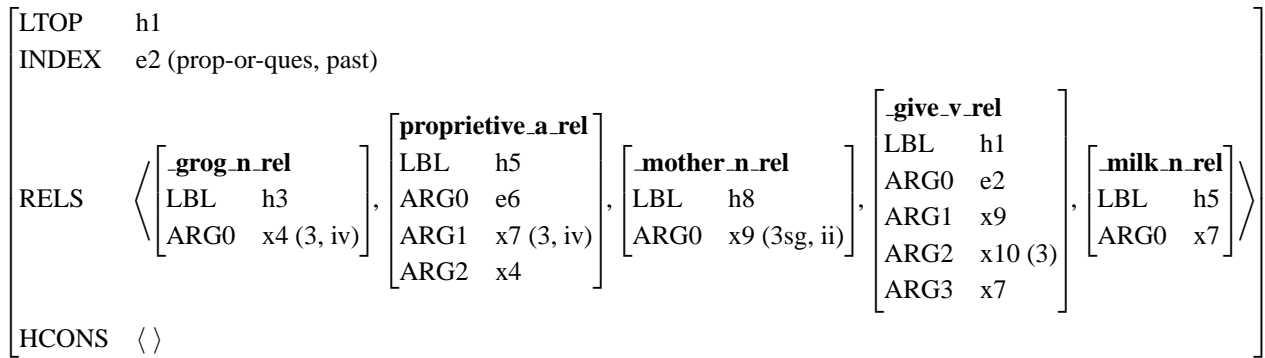


Figure 1: MRS for (1)

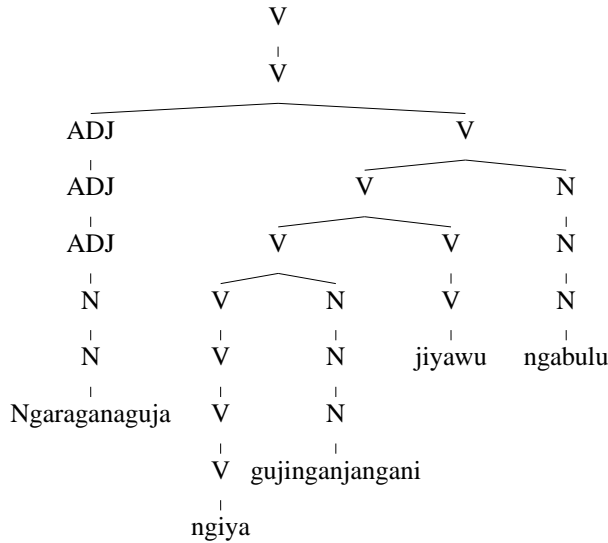
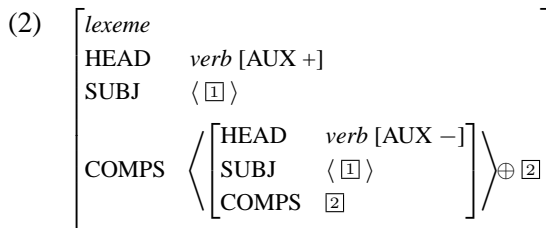
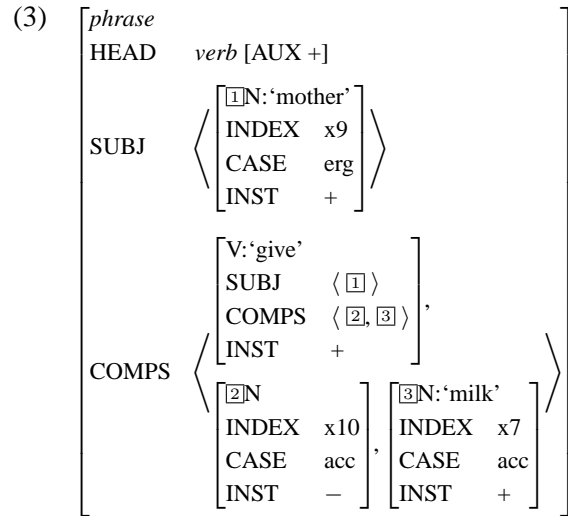


Figure 2: Phrase structure tree for (1)

the COMPS list. In this example, it first picks up the subject *gujunganjangani* (‘mother-ERG’), then the main verb *jiyawu* (‘give’), and then the object *ngabulu* (‘milk-ACC’).



The resulting V node over *ngiya gujunganjangani jiyawu ngabulu* is associated with the constraints sketched in (3).



Unlike in typical HPSG approaches, the information about the realized arguments is still exposed in the COMPS and SUBJ lists of this constituent.<sup>10</sup> This makes the necessary information available to separately-attaching modifiers (such as *ngaraganaguja* (‘grog-PROP.IV.ACC’)) so that they can check for case and number/gender compatibility and connect the semantic index of the argument they modify to a role in their own semantic contribution (in this case, the ARG1 of the ‘propriative’ relation).

### 3.4 Evaluation

The grammar was evaluated against a sample of naturally occurring data taken from one of the texts transcribed and translated by Nordlinger (1998) (‘The two Eaglehawks’, told by Molly Nurlanyma Grueman). Of the 92 sentences in this text, 20 overlapped with items in the development set, so the

<sup>10</sup>The feature INST, newly proposed for this analysis, records the fact that they have been instantiated by lexical heads.

	correct	parsed	unparsed	average
		incorrect		ambiguity
Existing vocab	50%	8%	42%	10.62
w/added vocab	<b>76%</b>	8%	14%	12.56

Table 1: Grammar performance on held-out data

evaluation was carried out only on the remaining 72 sentences. The evaluation was run twice: once with the grammar exactly as is, including the existing lexicon, and a second time after new lexical entries were added, using only existing lexical types. In some cases, the orthographic components of the lexical rules were also adjusted to accommodate the new lexical entries. In both test runs, the analyses of each test item were hand-checked against the translation provided by Nordlinger (1998). An item is counted as correctly analyzed if the set of analyses returned by the parser includes at least one with an MRS that matches the dependency structure, illocutionary force, tense, aspect, mood, person, number, and gender information indicated.

The results are shown in Table 1: With only lexical additions, the grammar was able to assign a correct parse to 55 (76%) of the test sentences, with an average ambiguity over these sentences of 12.56 parses/item.

### 3.5 Parse selection

The parsed portion of the development set (732 items) constitutes a sufficiently large corpus to train a parse selection model using the Redwoods disambiguation technology (Toutanova et al., 2005). As part of the grammar development process, the parses were annotated using the Redwoods parse selection tool (Oepen et al., 2004). The resulting treebank was used to select appropriate parameters by 10-fold cross-validation, applying the experimentation environment and feature templates of (Vellidal, 2007). The optimal feature set included 2-level grandparenting, 3-grams of lexical entry types, and both constituent weight features. In the cross-validation trials on the development set, this model achieved a parse selection accuracy of 80.2% (random choice baseline: 23.9%). A model with the same features was then trained on all 544 ambiguous examples

from the development set and used to rank the parses of the test set. It ranked the correct parse (exact match) highest in 75.0% of the test sentences. This is well above the random-choice baseline of 18.4%, and affirms the cross-linguistic validity of the parse-selection techniques.

### 3.6 Summary

This section has presented the Matrix-derived grammar of Wambaya, illustrating its semantic representations and analyses and measuring its performance against held-out data. I hope to have shown the grammar to be reasonably substantial, and thus an interesting case study with which to evaluate the Grammar Matrix itself.

## 4 Evaluation of Grammar Matrix

It is not possible to directly compare the development of a grammar for the same language, by the same grammar engineer, with and without the assistance of the Grammar Matrix. Therefore, in this section, I evaluate the usefulness of the Grammar Matrix by measuring the extent to which the Wambaya grammar as developed makes use of types defined in Matrix as well as the extent to which Matrix-defined types had to be modified. The former is in some sense a measure of the usefulness of the Matrix, and the latter is a measure of its correctness.

While the libraries and customization system were used in the initial grammar development, this evaluation primarily concerns itself with the Matrix core type hierarchy. The customization-provided Wambaya-specific type definitions for word order, lexical types, and coordination constructions were used for inspiration, but most needed fairly extensive modification. This is particularly unsurprising for basic word order, where the closest available option (“free word order”) was taken, in the absence of a pre-packaged analysis of non-configurationality and second-position phenomena. The other changes to the library output were largely side-effects of this fundamental difference.

Table 2 presents some measurements of the overall size of the Wambaya grammar. Since HPSG grammars consist of types organized into a hierarchy and instances of those types, the unit of measure for these evaluations will be types and/or instances. The

		N
Matrix types		891
ordinary	390	
pos disjunctions	591	
Wambaya-specific types		911
Phrase structure rules		83
Lexical rules		161
Lexical entries		1528

Table 2: Size of Wambaya grammar

	Matrix core types		w/ POS types	
Directly used	132	34%	136	15%
Indirectly used	98	25%	584	66%
Total types used	230	59%	720	81%
Types unused	160	41%	171	19%
Types modified	16	4%	16	2%
Total	390	100%	891	100%

Table 3: Matrix core types used in Wambaya grammar

Wambaya grammar includes 891 types defined in the Matrix core type hierarchy. These in turn include 390 ordinary types, and 591 ‘disjunctive’ types, the powerset of 9 part of speech types. These are provided in the Matrix so that Matrix users can easily refer to classes of, say, “nouns and verbs” or “nouns and verbs and adjectives”. The Wambaya-specific portion of the grammar includes 911 types. These types are invoked in the definitions of the phrase structure rules, lexical rules, and lexical entries.

Including the disjunctive part-of-speech types, just under half (49%) of the types in the grammar are provided by the Matrix. However, it is necessary to look more closely; just because a type is provided in the Matrix core hierarchy doesn’t mean that it is invoked by any rules or lexical entries of the Wambaya grammar. The breakdown of types used is given in Table 3. Types that are used directly are either called as supertypes for types defined in the Wambaya-specific portion of the grammar, or used as the value of some feature in a type constraint in the Wambaya-specific portion of the grammar. Types that are used indirectly are either ancestor types to types that are used directly, or types that are used as the value of a feature in a constraint in the Matrix core types on a type that is used (directly or indirectly) by the Wambaya-specific portion of the grammar.

Relatively few (16) of the Matrix-provided types needed to be modified. These were types that

were useful, but somehow unsuitable, and typically deeply interwoven into the type system, such that not using and then and defining parallel types in their place would be inconvenient.

Setting aside the types for part of speech disjunctions, 59% of the Matrix-provided types are invoked by the Wambaya-specific portion of the grammar. While further development of the Wambaya grammar might make use of some of the remaining 41% of the types, this work suggests that there is a substantial amount of information in the Matrix core type hierarchy which would better be stored as part of the typological libraries. In particular, the analyses of argument realization implemented in the Matrix were not used for this grammar. The types associated with argument realization in configurational languages should be moved into the word-order library, which should also be extended to include an analysis of Wambaya-style radical non-configurationality. At the same time, the lexical amalgamation analysis of the features used in long-distance dependencies (Sag, 1997) was found to be incompatible with the approach to argument realization in Wambaya, and a phrasal amalgamation analysis was implemented instead. This again suggests that lexical v. phrasal amalgamation should be encoded in the libraries, and selected according to the word order pattern of the language.

As for parts of speech, of the nine types provided by the Matrix, five were used in the Wambaya grammar (verb, noun, adj, adv, and det) and four were not (num, conj, comp, and adp(osition)). Four disjunctive types were directly invoked, to describe phenomena applying to nouns and adjectives, verbs and adverbs, anything but nouns, and anything but determiners. While it was convenient to have the disjunctive types predefined, it also seems that a much smaller set of types would suffice in this case. Since the nine proposed part of speech types have varying crosslinguistic validity (e.g., not all languages have conjunctions), it might be better to provide software support for creating the disjunctive types as the need arises, rather than predefining them.

Even though the number of Matrix-provided types is small compared to the grammar as a whole, the relatively short development time indicates that the types that were incorporated were quite useful. In providing the fundamental organization of the gram-

mar, to the extent that that organization is consistent with the language modeled, these types significantly ease the path to creating a working grammar.

The short development time required to create the Wambaya grammar presents a qualitative evaluation of the Grammar Matrix as a crosslinguistic resource, as one goal of the Grammar Matrix is to reduce the cost of developing precision grammars. The fact that a grammar capable of assigning valid analyses to an interesting portion of sentences from naturally occurring text could be developed in less than 5.5 person-weeks of effort suggests that this goal is indeed met. This is particularly encouraging in the case of endangered and other resource-poor languages. A grammar such as the one described here could be a significant aide in analyzing additional texts as they are collected, and in identifying constructions that have not yet been analyzed (cf. Baldwin et al, 2005).

## 5 Conclusion

This paper has presented a precision, hand-built grammar for the Australian language Wambaya, and through that grammar a case study evaluation of the LinGO Grammar Matrix. True validation of the Matrix *qua* hypothesized linguistic universals requires many more such case studies, but this first test is promising. Even though Wambaya is in some respects very different from the well-studied languages on which the Matrix is based, the existing machinery otherwise worked quite well, providing a significant jump-start to the grammar development process. While the Wambaya grammar has a long way to go to reach the complexity and range of linguistic phenomena handled by, for example, the LinGO English Resource Grammar, it was shown to provide analyses of an interesting portion of a naturally occurring text. This suggests that the methodology of building such grammars could be profitably incorporated into language documentation efforts.

The Grammar Matrix allows new grammars to directly leverage the expertise in grammar engineering gained in extensive work on previous grammars of better-studied languages. Furthermore, the design of the Matrix is such that it is not a static object, but intended to evolve and be refined as more languages are brought into its purview. Generalizing

the core hierarchy and libraries of the Matrix to support languages like Wambaya can extend its typological reach and further its development as an investigation in computational linguistic typology.

## Acknowledgments

I would like to thank Rachel Nordlinger for providing access to the data used in this work in electronic form, as well as for answering questions about Wambaya; Russ Hugo for data entry of the lexicon; Stephan Oepen for assistance with the parse ranking experiments; and Scott Drellishak, Stephan Oepen, and Laurie Poulson for general discussion. This material is based upon work supported by the National Science Foundation under Grant No. BCS-0644097.

## References

- J. Baldrige, S. Chatterjee, A. Palmer, and B. Wing. 2007. DotCCG and VisCCG: Wiki and programming paradigms for improved grammar engineering with OpenCCG. In T.H. King and E.M. Bender, editors, *GEAF 2007*, Stanford, CA. CSLI.
- T. Baldwin, J. Beavers, E.M. Bender, D. Flickinger, Ara Kim, and S. Oepen. 2005. Beauty and the beast: What running a broad-coverage precision grammar over the BNC taught us about the grammar — and the corpus. In S. Kepser and M. Reis, editors, *Linguistic Evidence: Empirical, Theoretical, and Computational Perspectives*, pages 49–70. Mouton de Gruyter, Berlin.
- J.A. Bateman, I. Kruijff-Korbayová, and G.-J. Kruijff. 2005. Multilingual resource sharing across both related and unrelated languages: An implemented, open-source framework for practical natural language generation. *Research on Language and Computation*, 3(2):191–219.
- E.M. Bender and D. Flickinger. 2005. Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core. In *IJCNLP-05 (Posters/Demos)*, Jeju Island, Korea.
- E.M. Bender, D. Flickinger, and S. Oepen. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In J. Carroll, N. Oostdijk, and R. Sutcliffe, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation, COLING 19*, pages 8–14, Taipei, Taiwan.
- E.M. Bender. 2007. Combining research and pedagogy in the development of a crosslinguistic grammar resource. In T.H. King and E.M. Bender, editors, *GEAF 2007*, Stanford, CA. CSLI.

- D.G. Bobrow, C. Condoravdi, R.S. Crouch, V. de Paiva, L. Karttunen, T.H. King, R. Nairn, L. Price, and A. Zaenen. 2007. Precision-focused textual inference. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Prague, Czech Republic.
- M. Butt, H. Dyvik, T.H. King, H. Masuichi, and C. Rohrer. 2002. The parallel grammar project. In J. Carroll, N. Oostdijk, and R. Sutcliffe, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at COLING 19*, pages 1–7.
- A. Copestake, D. Flickinger, C. Pollard, and I.A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language & Computation*, 3(2–3):281–332.
- A. Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI, Stanford, CA.
- C. Donohue and I.A. Sag. 1999. Domains in Warlpiri. Paper presented at HPSG 99, University of Edinburgh.
- S. Drellishak and E.M. Bender. 2005. A coordination module for a crosslinguistic grammar resource. In Stefan Müller, editor, *HPSG 2005*, pages 108–128, Stanford. CSLI.
- D. Flickinger and E.M. Bender. 2003. Compositional semantics in a multilingual grammar resource. In E.M. Bender, D. Flickinger, F. Fouvry, and M. Siegel, editors, *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, pages 33–42, Vienna, Austria.
- D. Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1):15–28.
- L. Hellan and P. Haugereid. 2003. NorSource: An exercise in Matrix grammar-building design. In E.M. Bender, D. Flickinger, F. Fouvry, and M. Siegel, editors, *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, pages 41–48, Vienna, Austria.
- R. Kim, M. Dalrymple, R.M. Kaplan, T.H. King, H. Masuichi, and T. Ohkuma. 2003. Multilingual grammar development via grammar porting. In E.M. Bender, D. Flickinger, F. Fouvry, and M. Siegel, editors, *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, pages 49–56, Vienna, Austria.
- T.H. King, M. Forst, J. Kuhn, and M. Butt. 2005. The feature space in parallel grammar writing. *Research on Language and Computation*, 3(2):139–163.
- A. Kinyon, O. Rambow, T. Scheffler, S.W. Yoon, and A.K. Joshi. 2006. The metagrammar goes multilingual: A cross-linguistic look at the V2-phenomenon. In *TAG+8*, Sydney, Australia.
- V. Kordoni and J. Neu. 2005. Deep analysis of Modern Greek. In K-Y Su, J. Tsujii, and J-H Lee, editors, *Lecture Notes in Computer Science*, volume 3248, pages 674–683. Springer-Verlag, Berlin.
- F. Lareau and L. Wanner. 2007. Towards a generic multilingual dependency grammar for text generation. In T.H. King and E.M. Bender, editors, *GEAF 2007*, pages 203–223, Stanford, CA. CSLI.
- J.T. Lønning and S. Oepen. 2006. Re-usable tools for precision machine translation. In *COLING|ACL 2006 Interactive Presentation Sessions*, pages 53–56, Sydney, Australia.
- M. Marimon, N. Bel, and N. Seghezzi. 2007. Test-suite construction for a Spanish grammar. In T.H. King and E.M. Bender, editors, *GEAF 2007*, Stanford, CA. CSLI.
- Stefan Müller. 2007. The Grammix CD-ROM: A software collection for developing typed feature structure grammars. In T.H. King and E.M. Bender, editors, *GEAF 2007*, Stanford, CA. CSLI.
- R. Nordlinger. 1998. *A Grammar of Wambaya, Northern Australia*. Research School of Pacific and Asian Studies, The Australian National University, Canberra.
- S. Oepen, E.M. Bender, U. Callmeier, D. Flickinger, and M. Siegel. 2002. Parallel distributed grammar engineering for practical applications. In *Proceedings of the Workshop on Grammar Engineering and Evaluation, COLING 19*, Taipei, Taiwan.
- S. Oepen, D. Flickinger, K. Toutanova, and C.D. Manning. 2004. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation*, 2(4):575–596.
- Stephan Oepen, Erik Velldal, Jan Tore Lønning, Paul Meurer, Victoria Rosn, and Dan Flickinger. 2007. Towards hybrid quality-oriented machine translation. On linguistics and probabilities in MT. In *TMI 2007*, Skvde, Sweden.
- C. Pollard and I.A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. CSLI, Stanford, CA.
- S. Robinson, G. Aumann, and S. Bird. 2007. Managing fieldwork data with Toolbox and the Natural Language Toolkit. *Language Documentation and Conservation*, 1:44–57.
- I.A. Sag. 1997. English relative clause constructions. *Journal of Linguistics*, 33(2):431–484.
- M. Siegel and E.M. Bender. 2002. Efficient deep processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization, COLING 19*, Taipei, Taiwan.
- K. Toutanova, C.D. Manning, D. Flickinger, and S. Oepen. 2005. Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation*, 3(1):83–105.
- E. Velldal. 2007. *Empirical Realization Ranking*. Ph.D. thesis, University of Oslo, Department of Informatics.

# Better Alignments = Better Translations?

**Kuzman Ganchev**  
Computer & Information Science  
University of Pennsylvania  
kuzman@cis.upenn.edu

**João V. Graça**  
L<sup>2</sup>F INESC-ID  
Lisboa, Portugal

**Ben Taskar**  
Computer & Information Science  
University of Pennsylvania  
taskar@cis.upenn.edu

## Abstract

Automatic word alignment is a key step in training statistical machine translation systems. Despite much recent work on word alignment methods, alignment accuracy increases often produce little or no improvements in machine translation quality. In this work we analyze a recently proposed agreement-constrained EM algorithm for unsupervised alignment models. We attempt to tease apart the effects that this simple but effective modification has on alignment precision and recall trade-offs, and how rare and common words are affected across several language pairs. We propose and extensively evaluate a simple method for using alignment models to produce alignments better-suited for phrase-based MT systems, and show significant gains (as measured by BLEU score) in end-to-end translation systems for six languages pairs used in recent MT competitions.

## 1 Introduction

The typical pipeline for a machine translation (MT) system starts with a parallel sentence-aligned corpus and proceeds to align the words in every sentence pair. The word alignment problem has received much recent attention, but improvements in standard measures of word alignment performance often do not result in better translations. Fraser and Marcu (2007) note that *none of the tens of papers published over the last five years has shown that significant decreases in alignment error rate (AER) result in significant increases in translation performance*. In this work, we show that by changing the way the word alignment models are trained and

used, we can get not only improvements in alignment performance, but also in the performance of the MT system that uses those alignments.

We present extensive experimental results evaluating a new training scheme for unsupervised word alignment models: an extension of the Expectation Maximization algorithm that allows effective injection of additional information about the desired alignments into the unsupervised training process. Examples of such information include “one word should not translate to many words” or that directional translation models should agree. The general framework for the extended EM algorithm with posterior constraints of this type was proposed by (Graça et al., 2008). Our contribution is a large scale evaluation of this methodology for word alignments, an investigation of how the produced alignments differ and how they can be used to consistently improve machine translation performance (as measured by BLEU score) across many languages on training corpora with up to hundred thousand sentences. In 10 out of 12 cases we improve BLEU score by at least  $\frac{1}{4}$  point and by more than 1 point in 4 out of 12 cases.

After presenting the models and the algorithm in Sections 2 and 3, in Section 4 we examine how the new alignments differ from standard models, and find that the new method consistently improves word alignment performance, measured either as alignment error rate or weighted F-score. Section 5 explores how the new alignments lead to consistent and significant improvement in a state of the art phrase base machine translation by using posterior decoding rather than Viterbi decoding. We propose a heuristic for tuning posterior decoding in the absence of annotated alignment data and show improvements over baseline systems for six different

language pairs used in recent MT competitions.

## 2 Statistical word alignment

Statistical word alignment (Brown et al., 1994) is the task identifying which words are translations of each other in a bilingual sentence corpus. Figure 2 shows two examples of word alignment of a sentence pair. Due to the ambiguity of the word alignment task, it is common to distinguish two kinds of alignments (Och and Ney, 2003). Sure alignments (S), represented in the figure as squares with borders, for single-word translations and possible alignments (P), represented in the figure as alignments without boxes, for translations that are either not exact or where several words in one language are translated to several words in the other language. Possible alignments can be used either to indicate optional alignments, such as the translation of an idiom, or disagreement between annotators. In the figure red/black dots indicates correct/incorrect predicted alignment points.

### 2.1 Baseline word alignment models

We focus on the hidden Markov model (HMM) for alignment proposed by (Vogel et al., 1996). This is a generalization of IBM models 1 and 2 (Brown et al., 1994), where the transition probabilities have a first-order Markov dependence rather than a zeroth-order dependence. The model is an HMM, where the hidden states take values from the source language words and generate target language words according to a translation table. The state transitions depend on the distance between the source language words. For source sentence  $s$  the probability of an alignment  $a$  and target sentence  $t$  can be expressed as:

$$p(\mathbf{t}, \mathbf{a} | \mathbf{s}) = \prod_j p_d(a_j | a_{j-1}) p_t(t_j | s_{a_j}), \quad (1)$$

where  $a_j$  is the index of the hidden state (source language index) generating the target language word at index  $j$ . As usual, a “null” word is added to the source sentence. Figure 1 illustrates the mapping between the usual HMM notation and the HMM alignment model.

### 2.2 Baseline training

All word alignment models we consider are normally trained using the Expectation Maximization

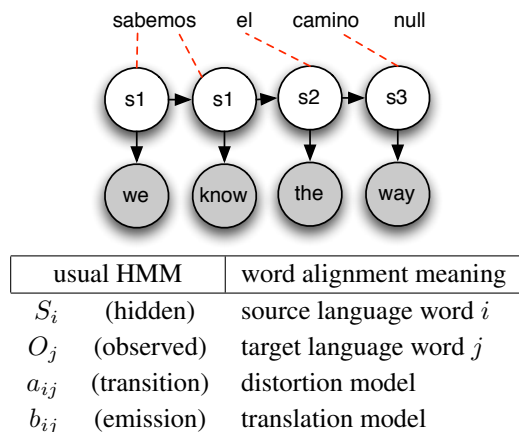


Figure 1: Illustration of an HMM for word alignment.

(EM) algorithm (Dempster et al., 1977). The EM algorithm attempts to maximize the marginal likelihood of the observed data ( $s, t$  pairs) by repeatedly finding a maximal lower bound on the likelihood and finding the maximal point of the lower bound. The lower bound is constructed by using posterior probabilities of the hidden alignments ( $a$ ) and can be optimized in closed form from expected sufficient statistics computed from the posteriors. For the HMM alignment model, these posteriors can be efficiently calculated by the Forward-Backward algorithm.

## 3 Adding agreement constraints

Graça et al. (2008) introduce an augmentation of the EM algorithm that uses constraints on posteriors to guide learning. Such constraints are useful for several reasons. As with any unsupervised induction method, there is no guarantee that the maximum likelihood parameters correspond to the intended meaning for the hidden variables, that is, more accurate alignments using the resulting model. Introducing additional constraints into the model often results in intractable decoding and search errors (e.g., IBM models 4+). The advantage of only constraining the posteriors during training is that the model remains simple while respecting more complex requirements. For example, constraints might include “one word should not translate to many words” or that translation is approximately symmetric.

The modification is to add a KL-projection step after the E-step of the EM algorithm. For each sentence pair instance  $\mathbf{x} = (s, t)$ , we find the posterior

distribution  $p_\theta(\mathbf{z}|\mathbf{x})$  (where  $\mathbf{z}$  are the alignments). In regular EM,  $p_\theta(\mathbf{z}|\mathbf{x})$  is used to complete the data and compute expected counts. Instead, we find the distribution  $q$  that is as close as possible to  $p_\theta(\mathbf{z}|\mathbf{x})$  in KL subject to constraints specified in terms of expected values of features  $\mathbf{f}(\mathbf{x}, \mathbf{z})$

$$\arg \min_q \text{KL}(q(\mathbf{z}) || p_\theta(\mathbf{z}|\mathbf{x})) \text{ s.t. } \mathbf{E}_q[\mathbf{f}(\mathbf{x}, \mathbf{z})] \leq \mathbf{b}. \quad (2)$$

The resulting distribution  $q$  is then used in place of  $p_\theta(\mathbf{z}|\mathbf{x})$  to compute sufficient statistics for the M-step. The algorithm converges to a local maximum of the log of the marginal likelihood,  $p_\theta(\mathbf{x}) = \sum_{\mathbf{z}} p_\theta(\mathbf{z}, \mathbf{x})$ , penalized by the KL distance of the posteriors  $p_\theta(\mathbf{z}|\mathbf{x})$  from the feasible set defined by the constraints (Graça et al., 2008):

$$\mathbf{E}_x[\log p_\theta(\mathbf{x}) - \min_{q: \mathbf{E}_q[\mathbf{f}(\mathbf{x}, \mathbf{z})] \leq \mathbf{b}} \text{KL}(q(\mathbf{z}) || p_\theta(\mathbf{z}|\mathbf{x}))],$$

where  $\mathbf{E}_x$  is expectation over the training data. They suggest how this framework can be used to encourage two word alignment models to agree during training. We elaborate on their description and provide details of implementation of the projection in Equation 2.

### 3.1 Agreement

Most MT systems train an alignment model in each direction and then heuristically combine their predictions. In contrast, Graça et al. encourage the models to agree by training them concurrently. The intuition is that the errors that the two models make are different and forcing them to agree rules out errors only made by one model. This is best exhibited in the rare word alignments, where one-sided “garbage-collection” phenomenon often occurs (Moore, 2004). This idea was previously proposed by (Matusov et al., 2004; Liang et al., 2006) although the the objectives differ.

In particular, consider a feature that takes on value 1 whenever source word  $i$  aligns to target word  $j$  in the forward model and -1 in the backward model. If this feature has expected value 0 under the mixture of the two models, then the forward model and backward model agree on how likely source word  $i$  is to align to target word  $j$ . More formally denote the forward model  $\vec{p}(\mathbf{z})$  and backward model  $\overleftarrow{p}(\mathbf{z})$  where  $\vec{p}(\mathbf{z}) = 0$  for  $\mathbf{z} \notin \vec{\mathbf{Z}}$  and  $\overleftarrow{p}(\mathbf{z}) = 0$  for  $\mathbf{z} \notin \overleftarrow{\mathbf{Z}}$  ( $\vec{\mathbf{Z}}$  and  $\overleftarrow{\mathbf{Z}}$  are possible forward and backward alignments). Define a mixture  $p(\mathbf{z}) = \frac{1}{2}\vec{p}(\mathbf{z}) + \frac{1}{2}\overleftarrow{p}(\mathbf{z})$

for  $\mathbf{z} \in \overleftarrow{\mathbf{Z}} \cup \vec{\mathbf{Z}}$ . Restating the constraints that enforce agreement in this setup:  $\mathbf{E}_q[\mathbf{f}(\mathbf{x}, \mathbf{z})] = \mathbf{0}$  with

$$f_{ij}(\mathbf{x}, \mathbf{z}) = \begin{cases} 1 & \mathbf{z} \in \vec{\mathbf{Z}} \text{ and } z_{ij} = 1 \\ -1 & \mathbf{z} \in \overleftarrow{\mathbf{Z}} \text{ and } z_{ij} = 1 \\ 0 & \text{otherwise} \end{cases}.$$

### 3.2 Implementation

EM training of hidden Markov models for word alignment is described elsewhere (Vogel et al., 1996), so we focus on the projection step:

$$\arg \min_q \text{KL}(q(\mathbf{z}) || p_\theta(\mathbf{z}|\mathbf{x})) \text{ s.t. } \mathbf{E}_q[\mathbf{f}(\mathbf{x}, \mathbf{z})] = \mathbf{0}. \quad (3)$$

The optimization problem in Equation 3 can be efficiently solved in its dual formulation:

$$\arg \min_\lambda \log \sum_{\mathbf{z}} p_\theta(\mathbf{z} | \mathbf{x}) \exp \{ \lambda^\top \mathbf{f}(\mathbf{x}, \mathbf{z}) \} \quad (4)$$

where we have solved for the primal variables  $q$  as:

$$q_\lambda(\mathbf{z}) = p_\theta(\mathbf{z} | \mathbf{x}) \exp \{ \lambda^\top \mathbf{f}(\mathbf{x}, \mathbf{z}) \} / Z, \quad (5)$$

with  $Z$  a normalization constant that ensures  $q$  sums to one. We have only one dual variable per constraint, and we optimize them by taking a few gradient steps. The partial derivative of the objective in Equation 4 with respect to feature  $i$  is simply  $\mathbf{E}_{q_\lambda}[\mathbf{f}_i(\mathbf{x}, \mathbf{z})]$ . So we have reduced the problem to computing expectations of our features under the model  $q$ . It turns out that for the agreement features, this reduces to computing expectations under the normal HMM model. To see this, we have by the definition of  $q_\lambda$  and  $p_\theta$ ,

$$\begin{aligned} q_\lambda(\mathbf{z}) &= \frac{\vec{p}(\mathbf{z} | \mathbf{x}) + \overleftarrow{p}(\mathbf{z} | \mathbf{x})}{2} \exp \{ \lambda^\top \mathbf{f}(\mathbf{x}, \mathbf{z}) \} / Z \\ &= \frac{\vec{q}(\mathbf{z}) + \overleftarrow{q}(\mathbf{z})}{2}. \end{aligned}$$

(To make the algorithm simpler, we have assumed that the expectation of the feature  $f_0(\mathbf{x}, \mathbf{z}) = \{1 \text{ if } \mathbf{z} \in \vec{\mathbf{Z}}; -1 \text{ if } \mathbf{z} \in \overleftarrow{\mathbf{Z}}\}$  is set to zero to ensure that the two models  $\vec{q}, \overleftarrow{q}$  are each properly normalized.) For  $\vec{q}$ , we have: ( $\overleftarrow{q}$  is analogous)

$$\begin{aligned} &\vec{p}(\mathbf{z} | \mathbf{x}) e^{\lambda^\top \mathbf{f}(\mathbf{x}, \mathbf{z})} \\ &= \prod_j \vec{p}_d(a_j | a_j - a_{j-1}) \vec{p}_t(\mathbf{t}_j | \mathbf{s}_{a_j}) \prod_{ij} e^{\lambda_{ij} f_{ij}(\mathbf{x}, z_{ij})} \\ &= \prod_{j, i=a_j} \vec{p}_d(i | i - a_{j-1}) \vec{p}_t(\mathbf{t}_j | \mathbf{s}_i) e^{\lambda_{ij} f_{ij}(\mathbf{x}, z_{ij})} \\ &= \prod_{j, i=a_j} \vec{p}_d(i | i - a_{j-1}) \vec{p}'_t(\mathbf{t}_j | \mathbf{s}_i). \end{aligned}$$



Where we have let  $\vec{p}'_t(\mathbf{t}_j|\mathbf{s}_i) = \vec{p}_t(\mathbf{t}_j|\mathbf{s}_i)e^{\lambda_{ij}}$ , and retained the same form for the model. The final projection step is detailed in Algorithm 1.

---

**Algorithm 1** AgreementProjection( $\vec{p}, \overleftarrow{p}$ )

---

- 1:  $\lambda_{ij} \leftarrow 0 \quad \forall i, j$
  - 2: **for** T iterations **do**
  - 3:  $\vec{p}'_t(j|i) \leftarrow \vec{p}_t(\mathbf{t}_j|\mathbf{s}_i)e^{\lambda_{ij}} \quad \forall i, j$
  - 4:  $\overleftarrow{p}'_t(i|j) \leftarrow \overleftarrow{p}_t(\mathbf{s}_i|\mathbf{t}_j)e^{-\lambda_{ij}} \quad \forall i, j$
  - 5:  $\vec{q} \leftarrow \text{forwardBackward}(\vec{p}'_t, \vec{p}_d)$
  - 6:  $\overleftarrow{q} \leftarrow \text{forwardBackward}(\overleftarrow{p}'_t, \overleftarrow{p}_d)$
  - 7:  $\lambda_{ij} \leftarrow \lambda_{ij} - \mathbf{E}_{\vec{q}}[a_i = j] + \mathbf{E}_{\overleftarrow{q}}[a_j = i] \quad \forall i, j$
  - 8: **end for**
  - 9: **return** ( $\vec{q}, \overleftarrow{q}$ )
- 

### 3.3 Decoding

After training, we want to extract a single alignment from the distribution over alignments allowable for the model. The standard way to do this is to find the most probable alignment, using the Viterbi algorithm. Another alternative is to use posterior decoding. In posterior decoding, we compute for each source word  $i$  and target word  $j$  the posterior probability under our model that  $i$  aligns to  $j$ . If that probability is greater than some threshold, then we include the point  $i - j$  in our final alignment. There are two main differences between posterior decoding and Viterbi decoding. First, posterior decoding can take better advantage of model uncertainty: when several likely alignment have high probability, posteriors accumulate confidence for the edges common to many good alignments. Viterbi, by contrast, must commit to one high-scoring alignment. Second, in posterior decoding, the probability that a

target word aligns to none or more than one word is much more flexible: it depends on the tuned threshold.

## 4 Word alignment results

We evaluated the agreement HMM model on two corpora for which hand-aligned data are widely available: the Hansards corpus (Och and Ney, 2000) of English/French parliamentary proceedings and the Europarl corpus (Koehn, 2002) with EPPS annotation (Lambert et al., 2005) of English/Spanish. Figure 2 shows two machine-generated alignments of a sentence pair. The black dots represent the machine alignments and the shading represents the human annotation (as described in the previous section), on the left using the regular HMM model and on the right using our agreement constraints. The figure illustrates a problem known as garbage collection (Brown et al., 1993), where rare source words tend to align to many target words, since the probability mass of the rare word translations can be hijacked to fit the sentence pair. Agreement constraints solve this problem, because forward and backward models cannot agree on the garbage collection solution.

Graça et al. (2008) show that alignment error rate (Och and Ney, 2003) can be improved with agreement constraints. Since AER is the standard metric for alignment quality, we reproduce their results using all the sentences of length at most 40. For the Hansards corpus we improve from 15.35 to 7.01 for the English  $\rightarrow$  French direction and from 14.45 to 6.80 for the reverse. For English  $\rightarrow$  Spanish we improve from 28.20 to 19.86 and from 27.54 to 19.18 for the reverse. These values are competitive with other state of the art systems (Liang et al., 2006).

Unfortunately, as was shown by Fraser and Marcu (2007) AER can have weak correlation with translation performance as measured by BLEU score (Papineni et al., 2002), when the alignments are used to train a phrase-based translation system. Consequently, in addition to AER, we focus on precision and recall.

Figure 3 shows the change in precision and recall with the amount of provided training data for the Hansards corpus. We see that agreement constraints improve both precision and recall when we

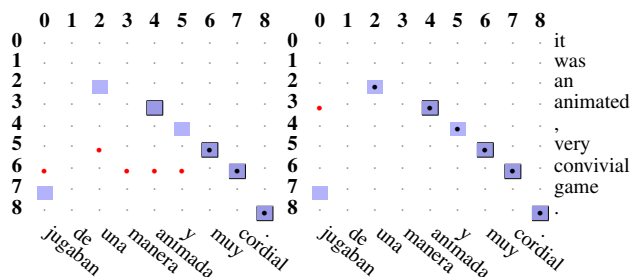


Figure 2: An example of the output of HMM trained on 100k the EPPS data. Left: Baseline training. Right: Using agreement constraints.

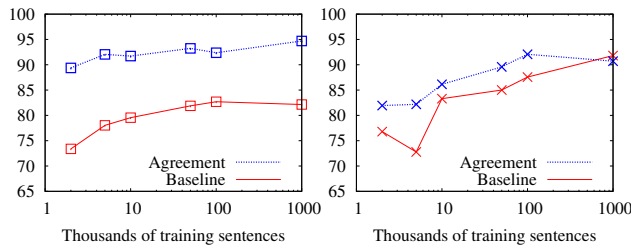


Figure 3: Effect of posterior constraints on precision (left) and recall (right) learning curves for Hansards En→Fr.

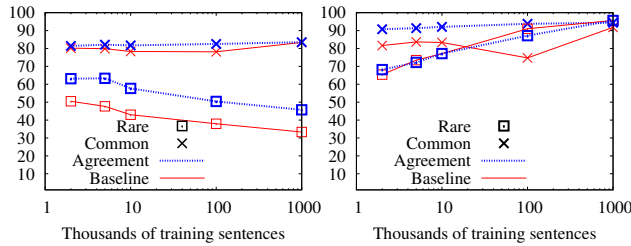


Figure 4: Left: Precision. Right: Recall. Learning curves for Hansards En→Fr split by rare (at most 5 occurrences) and common words.

use Viterbi decoding, with larger improvements for small amounts of training data. We see a similar improvement on the EPPS corpus.

Motivated by the garbage collection problem, we also analyze common and rare words separately. Figure 4 shows precision and recall learning curves for rare and common words. We see that agreement constraints improve precision but not recall of rare words and improve recall but not precision of common words.

As described above an alternative to Viterbi decoding is to accept all alignments that have probability above some threshold. By changing the threshold, we can trade off precision and recall. Figure 5 compares this tradeoff for the baseline and agreement model. We see that the precision/recall curve for agreement is entirely above the baseline curve, so for any recall value we can achieve higher precision than the baseline for either corpus. In Figure 6 we break down the same analysis into rare and non rare words.

Figure 7 shows an example of the same sentence, using the same model where in one case Viterbi decoding was used and in the other case Posterior decoding tuned to minimize AER on a development set

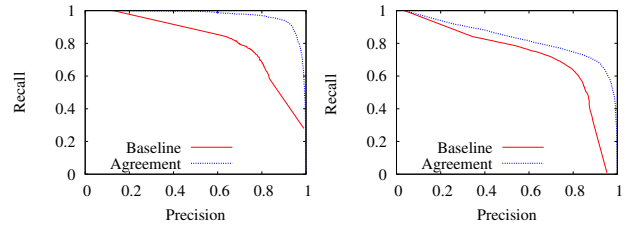


Figure 5: Precision and recall trade-off for posterior decoding with varying threshold. Left: Hansards En→Fr. Right: EPPS En→Es.

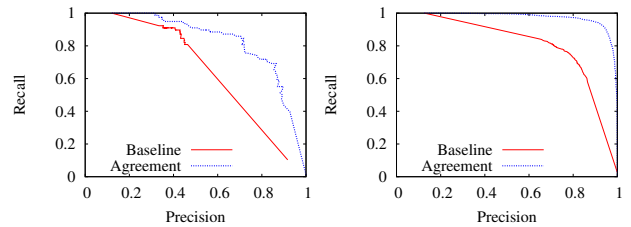


Figure 6: Precision and recall trade-off for posterior on Hansards En→Fr. Left: rare words only. Right: common words only.

was used. An interesting difference is that by using posterior decoding one can have n-n alignments as shown in the picture.

A natural question is how to tune the threshold in order to improve machine translation quality. In the next section we evaluate and compare the effects of the different alignments in a phrase based machine translation system.

## 5 Phrase-based machine translation

In this section we attempt to investigate whether our improved alignments produce improved machine

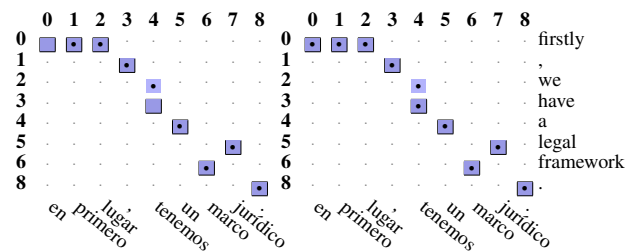


Figure 7: An example of the output of HMM trained on 100k the EPPS data using agreement HMM. Left: Viterbi decoding. Right: Posterior decoding tuned to minimize AER. The addition is *en-firstly* and *tenemos-have*.

translation. In particular we fix a state of the art machine translation system<sup>1</sup> and measure its performance when we vary the supplied word alignments. The baseline system uses GIZA model 4 alignments and the open source Moses phrase-based machine translation toolkit<sup>2</sup>, and performed close to the best at the competition last year.

For all experiments the experimental setup is as follows: we lowercase the corpora, and train language models from all available data. The reasoning behind this is that even if bilingual texts might be scarce in some domain, monolingual text should be relatively abundant. We then train the competing alignment models and compute competing alignments using different decoding schemes. For each alignment model and decoding type we train Moses and use MERT optimization to tune its parameters on a development set. Moses is trained using the grow-diag-final-and alignment symmetrization heuristic and using the default distance base distortion model. We report BLEU scores using a script available with the baseline system. The competing alignment models are GIZA Model 4, our implementation of the baseline HMM alignment and our agreement HMM. We would like to stress that the fair comparison is between the performance of the baseline HMM and the agreement HMM, since Model 4 is more complicated and can capture more structure. However, we will see that for moderate sized data the agreement HMM performs better than both its baseline and GIZA Model 4.

## 5.1 Corpora

In addition to the Hansards corpus and the Europarl English-Spanish corpus, we used four other corpora for the machine translation experiments. Table 1 summarizes some statistics of all corpora. The German and Finnish corpora are also from Europarl, while the Czech corpus contains news commentary. All three were used in recent ACL workshop shared tasks and are available online<sup>3</sup>. The Italian corpus consists of transcribed speech in the travel domain and was used in the 2007 workshop on spoken language translation<sup>4</sup>. We used the development and

<sup>1</sup>[www.statmt.org/wmt07/baseline.html](http://www.statmt.org/wmt07/baseline.html)

<sup>2</sup>[www.statmt.org/moses/](http://www.statmt.org/moses/)

<sup>3</sup><http://www.statmt.org>

<sup>4</sup><http://iwslt07.itc.it/>

Corpus	Train	Len	Test	Rare (%)		Unk (%)	
En, Fr	1018	17.4	1000	0.3	0.4	0.1	0.2
En, Es	126	21.0	2000	0.3	0.5	0.2	0.3
En, Fi	717	21.7	2000	0.4	2.5	0.2	1.8
En, De	883	21.5	2000	0.3	0.5	0.2	0.3
En, Cz	57	23.0	2007	2.3	6.6	1.3	3.9
En, It	20	9.4	500	3.1	6.2	1.4	2.9

Table 1: Statistics of the corpora used in MT evaluation. The training size is measured in thousands of sentences and Len refers to average (English) sentence length. Test is the number of sentences in the test set. Rare and Unk are the percentage of tokens in the test set that are rare and unknown in the training data, for each language.

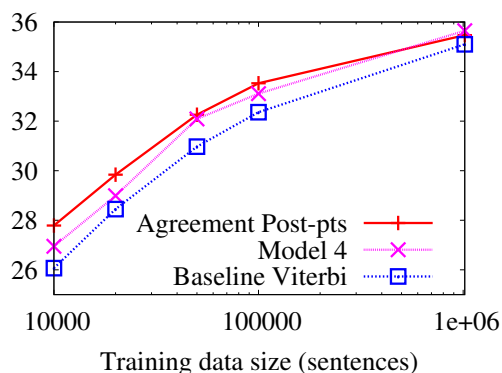


Figure 8: BLEU score as the amount of training data is increased on the Hansards corpus for the best decoding method for each alignment model.

tests sets from the workshops when available. For Italian corpus we used dev-set 1 as development and dev-set 2 as test. For Hansards we randomly chose 1000 and 500 sentences from test 1 and test 2 to be testing and development sets respectively.

Table 1 summarizes the size of the training corpus in thousands of sentences, the average length of the English sentences as well as the size of the testing corpus. We also report the percentage of tokens in the test corpus that are rare or not encountered in the training corpus.

## 5.2 Decoding

Our initial experiments with Viterbi decoding and posterior decoding showed that for our agreement model posterior decoding could provide better alignment quality. When labeled data is available, we can tune the threshold to minimize AER. When labeled data is not available we use a different heuristic to

tune the threshold: we choose a threshold that gives the same number of aligned points as Viterbi decoding produces. In principle, we would like to tune the threshold by optimizing BLEU score on a development set, but that is impractical for experiments with many pairs of languages. We call this heuristic posterior-points decoding. As we shall see, it performs well in practice.

### 5.3 Training data size

The HMM alignment models have a smaller parameter space than GIZA Model 4, and consequently we would expect that they would perform better when the amount of training data is limited. We found that this is generally the case, with the margin by which we beat model 4 slowly decreasing until a crossing point somewhere in the range of  $10^5$  -  $10^6$  sentences. We will see in section 5.3.1 that the Viterbi decoding performs best for the baseline HMM model, while posterior decoding performs best for our agreement HMM model. Figure 8 shows the BLEU score for the baseline HMM, our agreement model and GIZA Model 4 as we vary the amount of training data from  $10^4$  -  $10^6$  sentences. For all but the largest data sizes we outperform Model 4, with a greater margin at lower training data sizes. This trend continues as we lower the amount of training data further. We see a similar trend with other corpora.

#### 5.3.1 Small to Medium Training Sets

Our next set of experiments look at our performance in both directions across our 6 corpora, when we have small to moderate amounts of training data: for the language pairs with more than 100,000 sentences, we use only the first 100,000 sentences. Table 2 shows the performance of all systems on these datasets. In the table, post-pts and post-aer stand for posterior-points decoding and posterior decoding tuned for AER. With the notable exception of Czech and Italian, our system performs better than or comparable to both baselines, even though it uses a much more limited model than GIZA’s Model 4. The small corpora for which our models do not perform as well as GIZA are the ones with a lot of rare words. We suspect that the reason for this is that we do not implement smoothing, which has been shown to be important, especially in situations with a lot of rare words.

		X → En		En → X	
		Base	Agree	Base	Agree
De	GIZA M4	23.92		17.89	
	Viterbi	24.08	23.59	18.15	18.13
	post-pts	24.24	<b>24.65</b> <sup>(+)</sup>	18.18	<b>18.45</b> <sup>(+)</sup>
Fi	GIZA M4	18.29		11.05	
	Viterbi	18.79	18.38	11.17	11.54
	post-pts	18.88	<b>19.45</b> <sup>(++)</sup>	11.47	<b>12.48</b> <sup>(++)</sup>
Fr	GIZA M4	33.12		<b>26.90</b>	
	Viterbi	32.42	32.15	25.85	25.48
	post-pts	33.06	33.09 <sup>(≈)</sup>	25.94	26.54 <sup>(+)</sup>
	post-aer	31.81	<b>33.53</b> <sup>(+)</sup>	26.14	26.68 <sup>(+)</sup>
Es	GIZA M4	30.24		30.09	
	Viterbi	29.65	30.03	29.76	29.85
	post-pts	29.91	30.22 <sup>(++)</sup>	29.71	30.16 <sup>(+)</sup>
	post-aer	29.65	<b>30.34</b> <sup>(++)</sup>	29.78	<b>30.20</b> <sup>(+)</sup>
It	GIZA M4	51.66		<b>41.99</b>	
	Viterbi	<b>52.20</b>	52.09	41.40	41.28
	post-pts	51.06	51.14 <sup>(--)</sup>	41.63	41.79 <sup>(≈)</sup>
Cz	GIZA M4	<b>22.78</b>		<b>12.75</b>	
	Viterbi	21.25	21.89	12.23	12.33
	post-pts	21.37	22.51 <sup>(++)</sup>	12.16	12.47 <sup>(+)</sup>

Table 2: BLEU scores for all language pairs using up to 100k sentences. Results are after MERT optimization. The marks <sup>(++)</sup> and <sup>(+)</sup> denote that agreement with posterior decoding is better by 1 BLEU point and 0.25 BLEU points respectively than the best baseline HMM model; analogously for <sup>(--)</sup>, <sup>(-)</sup>; while <sup>(≈)</sup> denotes smaller differences.

#### 5.3.2 Larger Training Sets

For four of the corpora we have more than 100 thousand sentences. The performance of the systems on all the data is shown in Table 3. German is not included because MERT optimization did not complete in time. We see that even on over a million instances, our model sometimes performs better than GIZA model 4, and always performs better than the baseline HMM.

## 6 Conclusions

In this work we have evaluated agreement-constrained EM training for statistical word alignment models. We carefully studied its effects on word alignment recall and precision. Agreement training has a different effect on rare and common words, probably because it fixes different types of errors. It corrects the garbage collection problem for rare words, resulting in a higher precision. The recall improvement in common words

		X → En		En → X	
		Base	Agree	Base	Agree
Fi	GIZA M4	22.78		14.72	
	Viterbi	22.92	22.89	14.21	14.09
	post-pts	23.15	<b>23.43</b> (+)	14.57	<b>14.74</b> (≈)
Fr	GIZA M4	35.65		<b>31.15</b>	
	Viterbi	35.19	35.17	30.57	29.97
	post-pts	35.49	<b>35.95</b> (+)	29.78	30.02 (≈)
	post-aer	34.85	35.48 (+)	30.15	30.07 (≈)
Es	GIZA M4	31.62		<b>32.40</b>	
	Viterbi	31.75	31.84	31.17	31.09
	post-pts	31.88	32.19 (+)	31.16	31.56 (+)
	post-aer	31.93	<b>32.29</b> (+)	31.23	31.36 (≈)

Table 3: BLEU scores for all language pairs using all available data. Markings as in Table 2.

can be explained by the idea that ambiguous common words are different in the two languages, so the un-ambiguous choices in one direction can force the choice for the ambiguous ones in the other through agreement constraints.

To our knowledge this is the first extensive evaluation where improvements in alignment accuracy lead to improvements in machine translation performance. We tested this hypothesis on six different language pairs from three different domains, and found that the new alignment scheme not only performs better than the baseline, but also improves over a more complicated, intractable model. In order to get the best results, it appears that posterior decoding is required for the simplistic HMM alignment model. The success of posterior decoding using our simple threshold tuning heuristic is fortunate since no labeled alignment data are needed: Viterbi alignments provide a reasonable estimate of aligned words needed for phrase extraction. The nature of the complicated relationship between word alignments, the corresponding extracted phrases and the effects on the final MT system still begs for better explanations and metrics. We have investigated the distribution of phrase-sizes used in translation across systems and languages, following recent investigations (Ayan and Dorr, 2006), but unfortunately found no consistent correlation with BLEU improvement. Since the alignments we extracted were better according to all metrics we used, it should not be too surprising that they yield better translation performance, but perhaps a better trade-off can be achieved with a deeper understanding of

the link between alignments and translations.

## Acknowledgments

J. V. Graça was supported by a fellowship from Fundação para a Ciência e Tecnologia (SFRH/ BD/ 27528/ 2006). K. Ganchev was partially supported by NSF ITR EIA 0205448.

## References

- N. F. Ayan and B. J. Dorr. 2006. Going beyond AER: An extensive analysis of word alignments and their impact on MT. In *Proc. ACL*.
- P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, M. J. Goldsmith, J. Hajic, R. L. Mercer, and S. Mohanty. 1993. But dictionaries are data too. In *Proc. HLT*.
- P. F. Brown, S. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1994. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Royal Statistical Society, Ser. B*, 39(1):1–38.
- A. Fraser and D. Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Comput. Linguist.*, 33(3):293–303.
- J. Graça, K. Ganchev, and B. Taskar. 2008. Expectation maximization and posterior constraints. In *Proc. NIPS*.
- P. Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation.
- P. Lambert, A. De Gispert, R. Banchs, and J. B. Mariño. 2005. Guidelines for word alignment evaluation and manual alignment. In *Language Resources and Evaluation, Volume 39, Number 4*.
- P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proc. HLT-NAACL*.
- E. Matusov, Zens. R., and H. Ney. 2004. Symmetric word alignments for statistical machine translation. In *Proc. COLING*.
- R. C. Moore. 2004. Improving IBM word-alignment model 1. In *Proc. ACL*.
- F. J. Och and H. Ney. 2000. Improved statistical alignment models. In *ACL*.
- F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51.
- K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proc. ACL*.
- S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. COLING*.

# Mining Parenthetical Translations from the Web by Word Alignment

Dekang Lin

Google, Inc.  
Mountain View  
CA, 94043  
lindek@google.com

Shaojun Zhao<sup>†</sup>

University of Rochester  
Rochester  
NY, 14627  
zhao@cs.rochester.edu

Benjamin Van Durme<sup>†</sup>

University of Rochester  
Rochester  
NY, 14627  
vandurme@cs.rochester.edu

Marius Paşca

Google, Inc.  
Mountain View  
CA, 94043  
mars@google.com

## Abstract

Documents in languages such as Chinese, Japanese and Korean sometimes annotate terms with their translations in English inside a pair of parentheses. We present a method to extract such translations from a large collection of web documents by building a partially parallel corpus and use a word alignment algorithm to identify the terms being translated. The method is able to generalize across the translations for different terms and can reliably extract translations that occurred only once in the entire web. Our experiment on Chinese web pages produced more than 26 million pairs of translations, which is over two orders of magnitude more than previous results. We show that the addition of the extracted translation pairs as training data provides significant increase in the BLEU score for a statistical machine translation system.

## 1 Introduction

In natural language documents, a term (word or phrase) is sometimes followed by its translation in another language in a pair of parentheses. We call these **parenthetical translations**. The following examples are from Chinese web pages (we added underlines to indicate what is being translated):

- (1) 美国智库布鲁金斯学会 (Brookings Institution) 专研跨大西洋恐怖主义的美欧中心研究部主任杰若米·夏皮罗 (Jeremy Shapiro) 却认为, ...
- (2) 消化性溃疡的症状往往与消化不良 (indigestion), 胃炎 (gastritis) 等其他胃部疾病症状相似.
- (3) 殊不知美国是不会接受 (not going to fly) 这一想法的
- (4) ...当是一次式时, 叫线性规划 (linear programming).

<sup>†</sup>Contributions made during an internship at Google

The parenthetically translated terms are typically new words, technical terminologies, idioms, products, titles of movies, books, songs, and names of persons, organizations locations, *etc.* Commonly, an author might use such a parenthetical when a given term has no standard translation (or transliteration), and does not appear in conventional dictionaries. That is, an author might expect a term to be an *out-of-vocabulary* item for the target reader, and thus helpfully provides a reference translation *in situ*.

For example, in (1), the name *Shapiro* was transliterated as 夏皮罗. The name has many other transliterations in web documents, such as 夏皮洛, 夏比洛, 夏布洛, 夏皮羅, 沙皮罗, 夏皮若, 夏庇罗, 夏皮諾, 夏畢洛, 夏比羅, 夏比罗, 夏普羅, 夏批羅, 夏批罗, 夏彼羅, 夏彼罗, 夏培洛, 夏卜尔, 夏匹若 ..., where the three Chinese characters corresponds to the three syllables in Sha-pi-ro respectively. Each syllable may be mapped into different characters: 'Sha' into 夏 or 沙, 'pi' into 皮, 比, 批, and 'ro' into 罗, 洛, 若, ...

Variation is not limited to the effects of phonetic similarity. Story titles, for instance, are commonly translated semantically, often leading to a number of translations that have similar meaning, yet differ greatly in lexicographic form. For example, while the movie title *Syriana* is sometimes phonetically transliterated as 辛瑞那, 辛瑞纳, it may also be translated semantically according to the plot of the movie, e.g., 迷中迷 (mystery in mystery), 实录 (real log), 谍对谍 (spy against spy), 油激暗战 (oil-triggered secret war), 叙利亚 (Syria), 迷经 (mystery journey), ...

The parenthetical translations are extremely valuable both as a stand-alone on-line dictionary and as training data for statistical machine translation systems. They provide fresh data (new words) and cover a much wider range of topics than typical parallel training data for statistical machine translation systems.

The main contribution of this paper is a method for mining parenthetical translations by treating text snippets containing candidate pairs as a partially parallel corpus and using a word alignment algorithm to establish the correspondences between in-parenthesis and pre-parenthesis words.

This technique allows us to identify translation pairs even if they only appeared once on the entire web. As a result, we were able to obtain 26.7 million Chinese-English translation pairs from web documents in Chinese. This is over two orders of magnitude more than the number of extracted translation pairs in the previously reported results (Cao, *et al.* 2007).

The next section presents an overview of our algorithm, which is then detailed in Sections 3 and 4. We evaluate our results in Section 5 by comparison with bilingually linked Wikipedia titles and by using the extracted pairs as additional training data in a statistical machine translation system.

## 2 Mining Parenthetical Translations

A parenthetical translation matches the pattern:

$$(4) \quad f_1 f_2 \dots f_m (e_1 e_2 \dots e_n)$$

which is a sequence of  $m$  non-English words followed by a sequence of  $n$  English words in parentheses. In the remainder of the paper, we assume the non-English text is Chinese, but our technique works for other languages as well.

There have been two approaches to finding such parenthetical translations. One is to assume that the English term  $e_1 e_2 \dots e_n$  is given and use a search engine to retrieve text snippets containing  $e_1 e_2 \dots e_n$  from predominately non-English web pages (Nagata *et al.*, 2001, Kwok *et al.*, 2005). Another method (Cao *et al.*, 2007) is to go through a non-English corpus and collect all instances that match the parenthetical pattern in (4). We followed the second approach since it does not require a predefined list of English terms and is amendable for extraction at large scale.

In both cases, one can obtain a list of candidate pairs, where the translation of the in-parenthesis terms is a suffix of the pre-parenthesis text. The lengths and frequency counts of the suffixes have been used to determine what is the translation of the in-parenthesis term (Kwok *et al.*, 2005). For example, Table 1 lists a set of Chinese segments (with word-to-word translation underneath) that

precede the English term *Lower Egypt*. Owing to the frequency with which 下埃及 appears as a candidate, and in varying contexts, one has a good reason to believe 下埃及 is the correct translation of *Lower Egypt*.

... 下游 地区 为 下 埃及	downstream region is down Egypt
... 中心 位于 下 埃及	center located-at down Egypt
... 以及 所谓 的 下 埃及	and so-called of down Egypt
... 叫做 下 埃及	called down Egypt

Table 1: Chinese text preceding *Lower Egypt*

Unfortunately, this heuristic does not hold as often as one might imagine. Consider the candidates for *Channel Spacing* in Table 2. The suffix 间隔 (gap) has the highest frequency count. It is nonetheless an incomplete translation of *Channel Spacing*. The correct translations in rows  $c$  to  $h$  occurred with *Channel Spacing* only once.

$a$	... □ 为 频道 间隔	$\lambda$ is channel distance
$b$	... 其 频道 间隔	its channel distance
$c$	... 除了 降低 波道 间隔	in-addition-to reducing wave-passage distance
$d$	... 亦 展示 具 波道 间隔	also showed have wave-passage gap
$e$	... 也 就是 频道 间隔	also therefore is channel gap
$f$	... 且 频道 的 间隔	and channel 's gap
$g$	... 一个 重要 特性 是 信道 间隔	an important property is signal-passage gap
$h$	... 已经 能够 达到 通道 间隔	already able reach passage gap

Table 2: Text preceding *Channel Spacing*

The crucial observation we make here is that although the words like 信道 (in row  $g$ ) co-occurred with *Channel Spacing* only once, there are many co-occurrences of 信道 and *Channel* in other candidate pairs, such as:

- ... 而 不是 语音 信道 (Speech Channel)
- ... 块 平坦 衰落 信道 (Block Flat Fading Channel)
- ... 信道 B (Channel B)
- ... 光纤 信道 探针 (Fiber Channel Probes)

- ... 反向 信道 (Reverse Channel)
- ... 基带 滤波 反向 信道 (Reverse Channel)

Unlike previous approaches that rely solely on the preceding text of a single English term to determine its translation, we treat the entire collection of candidate pairs as a partially parallel corpus and establish the correspondences between the words using a word alignment algorithm.

At first glance, word alignment appears to be a more difficult problem than the extraction of parenthetical translations. Extraction of parenthetical translations need only determine the *first* pre-parenthesis word aligned with an in-parenthesis word, whereas word alignment requires the respective linking of *all* such (pre,in)-parenthesis word pairs. However, by casting the problem as word alignment, we are able to generalize across instances involving different in-parenthesis terms, giving us a larger number of, and more varied, example contexts per word.

For the examples in Table 2, the words 频道 (channel), 波道 (wave passage), 信道 (signal passage), and 通道 (passage) are aligned with *Channel*, and the words 间距 (distance) and 间隔 (gap) are aligned with *Spacing*. Given these alignments, the left boundary of the translated Chinese term is simply the leftmost word that is linked to one of the English words.

Our algorithm consists of two steps:

**Step 1** constructs a partially parallel corpus. This step takes as input a large collection of Chinese web pages and converts the sentences with parentheses containing English text into pairs of candidates.

**Step 2** uses an unsupervised algorithm to align English and Chinese and identify the term being translated according to the left-most aligned Chinese word. If no word alignments can be established, the pair is not considered a translation.

The next two sections present the details of each of the two steps.

### 3 Constructing a Partially Parallel Corpus

#### 3.1 Filtering out non-translations

The first step of our algorithm is to extract parentheticals and then filter out those that are not translations. This filtering is required as parenthetical translations represent only a small fraction of the

usages for parentheses (see Sec. 5.1). Table 3 shows some example of parentheses that are not translations.

The input to Step 1 is a collection of arbitrary web documents. We used the following criteria to identify candidate pairs:

- The pre-parenthesis text ( $T_p$ ) is predominantly in Chinese and the in-parenthesis text ( $T_i$ ) is predominantly in English.
- The concatenation of the digits in  $T_p$  must be identical to the concatenation of the digits in  $T_i$ . For example, rows *a*, *b* and *c* in Table 3 can be ruled out this way.
- If  $T_p$  contains some text in English, the same text must also appear in  $T_i$ . This filters out row *d*.
- Remove the pairs where  $T_i$  is part of anchor text. This rule is often applied to instances like row *e* where the file type tends to be inside a clickable link to a media file.
- The punctuation characters in  $T_p$  must also appear in  $T_i$ , unless they are quotation marks. The example in row *f* is ruled out because ‘/’ is not found in the pre-parenthesis text.

	Examples with translations in <i>italic</i>	Function of the in-parenthesis text
<i>a</i>	其数值通常在1.4~3.0之间 (MacArthur, 1967) <i>The range of its values is within 1.4~3.0 (MacArthur, 1967)</i>	to provide citation
<i>b</i>	越航北京/胡志明 (VN901 15:20-22:30) <i>Vietnam Airlines Beijing/Ho Chi Minh (VN901 15:20-22:30)</i>	flight information
<i>c</i>	销售台球桌 (255-8FT) <i>sale of pool table (255-8FT)</i>	product Id.
<i>d</i>	// 主程序 // void main ( void ) <i>// main program // void main (void )</i>	function declaration
<i>e</i>	电影名称: 千年湖 (DVD) <i>movie title: Thousand Year Lake (DVD)</i>	DVD is the file type
<i>f</i>	水样 所 消耗 的 质量 ( g/L) <i>mass consumed by water sample (g/L)</i>	measurement unit
<i>g</i>	柔和保养面油 (Sensitive) <i>gentle protective facial cream (Sensitive)</i>	to indicate the type of the cream
<i>h</i>	美国九大搜索引擎评测第四章 (Ask Jeeves) <i>Evaluation of Nine Main Search Engines in the US: Chapter 4 (Ask Jeeves)</i>	Chapter 4 is about Ask Jeeves

Table 3: Other uses of parentheses



The instances in rows  $g$  and  $h$  cannot be eliminated by these simple rules, and are filtered only later, as we fail to discover a convincing word alignment.

### 3.2 Constraining term boundaries

Similar to (Cao *et al.* 2007), we segmented the pre-parenthesis Chinese text and restrict the term boundary to be one of the segmentation boundaries. Since parenthetical translations are mostly translation of terms, it makes sense to further constrain the left boundary of the Chinese side to be a term boundary. Determining what should be counted as a term is a difficult task and there are not yet well-accepted solutions (Sag *et al.* 2003).

We compiled an approximate term vocabulary by taking the top 5 million most frequent Chinese queries as according to a fully anonymized collection of search engine query logs.

Given a Chinese sentence, we first identify all (possibly overlapping) sequences of words in the sentence that match one of the top-5M queries. A matching sequence is called a maximal match if it is not properly contained in another matching sequence. We then define the **potential boundary positions** to be the boundaries of maximal matches or words that are not covered by any of the top-5M queries.

### 3.3 Length-based trimming

If there are numerous Chinese words preceding a pair of parentheses containing two English words, it is very unlikely for all but the right-most few Chinese words to be part of the translation of the English words. Including extremely long sequences as potential candidates introduces significantly more noise and makes word alignment harder than necessary. We therefore trimmed the pre-parenthesis text with a length-based constraint. The cut-off point is the first (counting from right to left) potential boundary position (see Sec. 3.2) such that  $C \geq 2E + K$ , where  $C$  is the length of the Chinese text,  $E$  is the length of the English text in the parentheses and  $K$  is a constant (we used  $K=6$  in our experiments). The lengths  $C$  and  $E$  are measured in bytes, except when the English text is an abbreviation (in that case,  $E$  is multiplied by 5).

## 4 Word Alignment

Word alignment is a well-studied topic in Machine Translation with many algorithms having been

proposed (Brown *et al.* 1993; Och and Ney 2003). We used a modified version of one of the simplest word alignment algorithms called Competitive Linking (Melamed, 2000). The algorithm assumes that there is a score associated with each pair of words in a bi-text. It sorts the word pairs in descending order of their scores, selecting pairs based on the resultant order. A pair of words is linked if none of the two words were previously linked to any other words. The algorithm terminates when there are no more links to make.

Tiedemann (2004) compared a variety of alignment algorithms and found Competitive Linking to have one of the highest precision scores. A disadvantage of Competitive Linking, however, is that the alignments are restricted word-to-word alignments, which implies that multi-word expressions can only be partially linked at best.

### 4.1 Dealing with multi-word alignment

We made a small change to Competitive Linking to allow consecutive sequence of words on one side to be linked to the same word on the other side. Specifically, instead of requiring both  $e_i$  and  $f_j$  to have no previous linkages, we only require that at least one of them be unlinked and that (suppose  $e_i$  is unlinked and  $f_j$  is linked to  $e_k$ ) none of the words between  $e_i$  and  $e_k$  be linked to any word other than  $f_j$ .

### 4.2 Link scoring

We used  $\varphi^2$  (Gale and Church, 1991) as the link score in the modified competitive linking algorithm, although there are many other possible choices for the link scores, such as  $\chi^2$  (Zhang, S. Vogel. 2005), log-likelihood ratio (Dunning, 1993) and discriminatively trained weights (Taskar *et al.*, 2005). The  $\varphi^2$  statistics for a pair of words  $e_i$  and  $f_j$  is computed as

$$\varphi^2 = \frac{(ad - bc)^2}{(a+b)(a+c)(b+d)(c+d)}$$

where

$a$  is the number of sentence pairs containing both  $e_i$  and  $f_j$ ;

$a+b$  is the number of sentence pairs containing  $e_i$ ;

$a+c$  is the number of sentence pairs containing  $f_j$ ;

$d$  is the number of sentence pairs containing neither  $e_i$  nor  $f_j$ .

The  $\phi^2$  score ranges from 0 to 1. We set a threshold at 0.001, below which the  $\phi^2$  scores are treated as 0.

### 4.3 Bias in the partially parallel corpus

Since only the last few Chinese words in a candidate pair are expected to be translated, there should be a preference for linking the words towards the end of the Chinese text. One advantage of Competitive Linking is that it is quite easy to introduce such preferences into the algorithm, by using the word positions to break ties of the  $\phi^2$  scores when sorting the word pairs.

### 4.4 Capturing syllable-level regularities

Many of the parenthetical translations involve proper names, which are often transliterated according to the sound. Word alignment algorithms have generally ignored syllable-level regularities in transliterated terms. Consider again the *Shapiro* example in the introduction section. There are numerous correct transliterations for the same English word, some of which are not very frequent. For example, the word 夏布洛 happens to have a similar  $\phi^2$  score with *Shapiro* as the word 流利 (fluency), which is totally unrelated to *Shapiro* but happened to have the same co-occurrence statistics in the (partially) parallel corpus.

Previous approaches to parenthetical translations relied on specialized algorithms to deal with transliterations (Cao *et al*, 2007; Jiang *et al*, 2007; Wu and Chang, 2007). They convert Chinese words into their phonetic representations (Pinyin) and use the known transliterations in a bilingual dictionary to train a transliteration model.

We adopted a simpler approach that does not require any additional resources such as pronunciation dictionaries and bilingual dictionaries. In addition to computing the  $\phi^2$  scores between words, we also compute the  $\phi^2$  scores of prefixes and suffixes of Chinese and English words. For both languages, the prefix of a word is defined as the first three bytes of the word and the suffix is defined as the last three bytes. Since we used UTF-8 encoding, the first and last three bytes of a Chinese word, except in very rare cases, correspond to the first and last Chinese character of the word. Table 4 lists the English prefixes and suffixes that have the highest  $\phi^2$  scores with the Chinese prefix 夏 and suffix 洛.

Type	Chinese	English
prefix	夏	sha, amo, cha, sum, haw, lav, lun, xia, xal, hnl, shy, eve, she, cfh, ...
suffix	洛	rlo, llo, ouh, low, ilo, owe, lol, lor, zlo, klo, gue, ude, vir, row, oro, olo, aro, ulo, ero, iro, rro, loh, lok, ...

Table 4: Example prefixes and suffixes with top  $\phi^2$

In our modified version of the competitive linking algorithm, the link score of a pair of words is the sum of the  $\phi^2$  scores of the words themselves, their prefixes and their suffixes.

In addition to syllable-level correspondences in transliterations, the  $\phi^2$  scores of prefixes and suffixes can also capture correlations in morphologically composed words. For example, the Chinese prefix 三 (three) has a relatively high  $\phi^2$  score with the English prefix *tri*. Such scores enable word alignments to be made that may otherwise be missed. Consider the following text snippet:

..... 三 嗒 氟草胺 (triaziflam)

The correct translation for *triaziflam* is 三嗒氟草胺. However, the Chinese term is segmented as 三 + 嗒 + 氟草胺. The association between 三 (three) and *triaziflam* is very weak because 三 is a very frequent word, whereas *triaziflam* is an extremely rare word. With the addition of the  $\phi^2$  score between 三 and *tri*, we were able to correctly establish the connection between *triaziflam* and 三.

It turns out to be quite effective to assume prefixes and suffixes of words consist of three bytes, despite its apparent simplicity. The benefit of  $\phi^2$  scores for prefixes and suffixes is not limited to morphemes that happen to be three bytes long. For example, the English morpheme “du-” corresponds to the Chinese character 二 (two). Although the  $\phi^2$  between *du* and 二 won’t be computed, we do find high  $\phi^2$  scores between 二 and *due* and between 二 and *dua*. The three letter prefixes account for many of the words with the *du-* prefix.

## 5 Experimental Results

We extracted from Chinese web pages about 1.58 billion unique sentences with parentheses that contain ASCII text. We removed duplicate sentences so that duplications of web documents will not skew the statistics. By applying the filtering algorithm in Sec. 3.1, we constructed a partially paral-

lel corpus with 126,612,447 candidate pairs (46,791,841 unique), which is about 8% of the number of sentences. Using the word alignment algorithm in Sec. 4, we extracted 26,753,972 translation pairs between 13,471,221 unique English terms and 11,577,206 unique Chinese terms.

Parenthetical translations mined from the Web have mostly been evaluated by manual examination of a small sample of results (usually a few hundred entries) or in a Cross Lingual Information Retrieval setup. There does not yet exist a common evaluation data set.

### 5.1 Evaluation with Wikipedia

Our first evaluation is based on translations in Wikipedia, which contains far more terminology and proper names than bilingual dictionaries. We extracted the titles of Chinese and English Wikipedia articles that are linked to each other and treated them as gold standard translations. There are 79,714 such pairs. We removed the following types of pairs because they are not translations or are not terms:

- Pairs with identical strings. For example, both English and Chinese versions have an entry titled “.ch”;
- Pairs where the English term begins with a digit, e.g., “245”, “300 BC”, “1991 in film”;
- Pairs where the English term matches the regular expression ‘List of .\*’, e.g., “List of birds”, “List of cinemas in Hong Kong”;
- Pairs where the Chinese title does not have any non-ASCII code. For example, the English entry “SynCFusion” is linked to “.NET Framework” in the Chinese Wikipedia.

The resulting data set contains 68,131 translation pairs between 62,581 Chinese terms and 67,613 English terms. Only a small percentage of terms have more than one translation. Whenever there is more than one translation, we randomly pick one as the answer key.

For each Chinese and English word in the Wikipedia data, we first find whether there is a translation for the word in the extracted translation pairs. The **Coverage** of the Wikipedia data is measured by the percentage of words for which one or more translations are found. We then see whether our most frequent translation is an **Exact Match** of the answer key in the Wikipedia data.

	Coverage	Exact Match
<b>Full</b>	<b>70.8%</b>	<b>36.4%</b>
<b>-term</b>	67.1%	34.8%
<b>-pre-suffix</b>	67.6%	34.4%
<b>IBM</b>	67.6%	31.2%
<b>LDC</b>	10.8%	4.8%

Table 5: Chinese to English Results

	Coverage	Exact Match
<b>Full</b>	<b>59.6%</b>	<b>27.9%</b>
<b>-term</b>	<b>59.6%</b>	27.5%
<b>-pre-suffix</b>	58.9%	27.4%
<b>IBM</b>	52.4%	13.4%
<b>LDC</b>	3.0%	1.4%

Table 6: English to Chinese Results

Table 5 and 6 show the Chinese-to-English and English-to-Chinese results for the following systems:

**Full** refers to our system described in Sec. 3 and 4;

**-term** is the system without the use of query logs to restrict potential term boundary positions (Sec. 3.2);

**-pre-suffix** is the system without using the  $\phi^2$  score of the prefixes and suffixes;

**IBM** refers to a system where we substitute our word alignment algorithm with IBM Model 1 and Model 2 followed by the HMM alignment (Och and Ney 2003), which is a common configuration for the word alignment components in machine translations systems;

**LDC** refers to the LDC2.0 English to Chinese bilingual dictionary with 161,117 translation pairs.

It can be seen that the use of queries to constrain boundary positions and the addition of  $\phi^2$  scores of prefixes/suffixes improve the percentage of Exact Match. The IBM Model tends to make many more alignments than Complete Linking. While this is often beneficial for machine translation systems, it is not very suitable for creating bilingual dictionaries, where precision is of paramount importance. The LDC dictionary was manually compiled from diverse resources within LDC and (mostly) from the Internet. Its coverage of Wikipedia data is extremely low, compared to our method.

English	Wikipedia Translation	Parenthetical Translation
Pumping lemma	泵引理	引理 <sup>1</sup>
Topic-prominent language	话题优先语言	突出性语言 <sup>1</sup>
Yoido Full Gospel Church	汝矣岛纯福音教会	全备福音教会 <sup>1</sup>
First Bulgarian Empire	第一保加利亚帝国	强大的保加利亚帝国 <sup>2</sup>
Vespid	黄蜂	针对境内胡蜂 <sup>2</sup>
Ibrahim Rugova	易卜拉欣·鲁戈瓦	鲁戈瓦 <sup>3</sup>
Jerry West	杰里·韦斯特	威斯特 <sup>3</sup>
Nicky Butt	尼基·巴特	巴特 <sup>3</sup>
Benito Mussolini	贝尼托·墨索里尼	墨索里尼 <sup>3</sup>
Ecology of Hong Kong	香港生态	本文介绍的*
Paracetamol	对乙酰氨基酚	扑热息痛*
Thermidor	热月	必杀*
Udo	独活	乌多
Public opinion	舆论	公众舆论
Michael Bay	麦可·贝	迈克尔·贝
Dagestan	达吉斯坦共和国	达吉斯坦
Battle of Leyte Gulf	莱特湾海战	莱伊特海湾战役
Glock	格洛克手枪	格洛克
Ergonomics	人因工程学	工效学
Frank Sinatra	法兰·仙纳杜拉	法兰克辛纳屈
Zaragoza	萨拉戈萨省	萨拉戈萨
Komodo	科莫多岛	科摩多岛
Eli Vance	伊莱·万斯	伊莱·凡斯博士
Manitoba	缅尼托巴	曼尼托巴省
Giant Bottlenose Whale	阿氏贝喙鲸	巨瓶鼻鲸
Exclusionary rule	证据排除法则	证据排除规则
Computer worm	蠕虫病毒	计算机蠕虫
Social network	社会性网络	社会网络
Glasgow School of Art	格拉斯哥艺术学校	格拉斯哥艺术学院
Dee Hock	狄伊·哈克	迪伊·霍克
Bondage	绑缚	束缚
The China Post	英文中国邮报	中国邮报
Rachel	拉结	瑞秋
John Nash	约翰·纳西	约翰·纳什
Hattusa	哈图沙	哈图萨
Bangladesh	孟加拉国	孟加拉

Table 7: A random sample of non-exact-matches

<sup>1</sup>the extracted translation is too short

<sup>2</sup>the extracted translation is too long

<sup>3</sup>the extracted translation contains only the last name

\*the extracted term is completely wrong.

Note that Exact Match is a rather stringent criterion. Table 7 shows a random sample of extracted parenthetical translations that failed the Exact Match test. Only a small percentage of them are genuine errors. We nonetheless adopted this measure because it has the advantage of automated evaluation and our goal is mainly to compare the relative performances.

To determine the upper bound of the coverage of our web data, for each Wikipedia English term we searched within the total set of available parenthesized text fragments (our English candidate set before filtering as by Step 1). We discovered 81% of the Wikipedia titles, which is approximately 10% above the coverage of our final output. This indicates a minor loss of recall because of mistakes made in filtering (Sec. 3.1) and/or word alignment.

## 5.2 Evaluation with term translation requests

To evaluate the coverage of output produced by their method, Cao *et al* (2007) extracted English queries from the query log of a Chinese search engine. They assume that the reason why users typed the English queries in a Chinese search box is mostly to find out their Chinese translations. Examining our own Chinese query logs, however, the most-frequent English queries appear to be navigational queries instead of translation requests. We therefore used the following regular expression to identify queries that are unambiguously translation requests:

`/^[a-zA-Z]*的中文$/`

where的中文means “’s Chinese”. This regular expression matched 1579 unique queries in the logs. We manually judged the translation for 200 of them. A small random sample of the 200 is shown in Table 8. The empty cells indicate that the English term is missing from our translation pairs. We use \* to mark incorrect translations. When compared with the sample queries in (Cao *et al.*, 2007), the queries in our sample seem to contain more phrasal words and technical terminology. It is interesting to see that even though parenthetical translations tend to be out-of-vocabulary words, as we have remarked in the introduction, the sheer size of the web means that occasionally translations of common words such as ‘use’ are sometimes included as well.

buckingham palace	白金汉宫
chinadaily	中国日报
coo	首席运营官
diammonium sulfate	
emilio pucci	埃米里奥·普奇
finishing school	精修学校
gloria	格洛丽亚
horny	长角收割者*
jam	詹姆
lean six sigma	精益六西格玛
meiosis	减数分裂
near miss	迹近错失
pachycephalosaurus	肿头龙
pops	持久性有机污染物
recreation vehicle	休闲露营车
shanghai ethylene cracker complex	
stenonychosaurus	细爪龙
theanine	茶氨酸
use	使用
with you all the time	回想和你在一起的日子里

Table 8: A small sample of manually judged query translations

We compared our results with translations obtained from Google and Yahoo’s translation services. The numbers of correct translations for the random sample of 200 queries are as follows:

Systems	Google	Yahoo!	Mined	Mined+G
Correct	115	84	116	135

Our system’s outputs (**Mined**) have the same accuracy as the Google Translate. Our outputs have results for 154 out of the 200 queries. The 46 missing results are considered incorrect. If we combine our results with Google Translate by looking up Google results for missing entries, the accuracy increases from 56% to 68% (**Mined+G**). If we treat the LDC Chinese-English Dictionary 2.0 as a translator, it only covers 20.5% of the 200 queries.

### 5.3 Evaluation with SMT

The extracted translations may serve as training data for statistical machine translation systems. To evaluate their effectiveness for this purpose, we trained a baseline phrase-based SMT system (Koehn *et al.*, 2003; Brants *et al.*, 2007) with the FBIS Chinese-English parallel text (NIST, 2003). We then added the extracted translation pairs as

additional parallel training corpus. This resulted in a 0.57 increase of BLEU score based on the test data in the 2006 NIST MT Evaluation Workshop.

## 6 Related Work

Nagata *et al.* (2001) made the first proposal to mine translations from the web. Their work was concentrated on terminologies, and assumed the English terms were given as input. Wu and Chang (2007), Kwok *et al.* (2005) also employed search engines and assumed the English term given as input, but their focus was on name transliteration. It is difficult to build a truly large-scale translation lexicon this way because the English terms themselves may be hard to come by.

Cao *et al.* (2007), like us, used a 300GB collection of web documents as input. They used supervised learning to build models that deal with phonetic transliterations and semantic translations separately. Our work relies on unsupervised learning and does not make a distinction between translations and transliterations. Furthermore, we are able to extract two orders of magnitude more translations from than (Cao *et al.*, 2007).

## 7 Conclusion

We presented a method to apply a word alignment algorithm on a partially parallel corpus to extract translation pairs from the web. Treating the translation extraction problem as a word alignment problem allowed us to generalize across instances involving different in-parenthesis terms. Our algorithm extends Competitive Linking to deal with multi-word alignments and takes advantage of word-internal correspondences between transliterated words or morphologically composed words. Finally, through our discussion of parallel Wikipedia topic titles as a gold standard, we presented the first evaluation of such an extraction system that went beyond manual judgments on small sized samples.

## Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments.

## References

- T. Brants, A. Popat, P. Xu, F. Och and J. Dean, *Large Language Models for Machine Translation*, EMNLP-CoNLL-2007.
- P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, and R.L. Mercer. 1993. *The mathematics of statistical machine translation: Parameter estimation*. *Computational Linguistics*, 19(2):263–311.
- G. Cao, J. Gao and J.Y. Nie. 2007. *A system to mine large-scale bilingual dictionaries from monolingual Web pages*, MT Summit, pp. 57-64.
- T. Dunning. 1993. *Accurate Methods for the Statistics of Surprise and Coincidence*. *Computational Linguistics* 19, 1.
- W. Gale and K. Church. 1991. *Identifying word correspondence in parallel text*. In *Proceedings of the DARPA NLP Workshop*.
- L. Jiang, M. Zhou, L.F. Chien, C. Niu. 2007. *Named Entity Translation with Web Mining and Transliteration*. In *Proc. of IJCAI-2007*. pp. 1629-1634.
- P. Koehn, F. Och and D. Marcu, *Statistical Phrase-based Translation*, In *Proc. of HLT-NAACL 2003*.
- K.L. Kwok, P. Deng, N. Dinstl, H.L. Sun, W. Xu, P. Peng, and J. Doyon. 2005. *CHINET: a Chinese name finder system for document triage*. In *Proceedings of 2005 International Conference on Intelligence Analysis*.
- I.D. Melamed. 2000. *Models of translational equivalence among words*. *Computational Linguistics*, 26(2):221–249.
- M. Nagata, T. Saito, and K. Suzuki. 2001. *Using the Web as a bilingual dictionary*. In *Proc. of ACL 2001 DD-MT Workshop*, pp.95-102.
- NIST. 2003. The NIST machine translation evaluations. <http://www.nist.gov/speech/tests/mt/>.
- F.J. Och and H. Ney. 2003. *A systematic comparison of various statistical alignment models*. *Computational Linguistics*, 29(1):19–51.
- I.A. Sag, T. Baldwin, F. Bond, A. Copestake, and D. Flickinger. 2002. *Multiword expressions: A pain in the neck for NLP*. In *Proc. of CICLing-2002*, pp 1–15, Mexico City, Mexico.
- B. Taskar, S. Lacoste-Julien, and D. Klein. 2005. *A discriminative matching approach to word alignment*. In *Proc. of HLT/EMNLP-05*. Vancouver, BC.
- J. Tiedemann. 2004. *Word to word alignment strategies*. In *Proceedings of the 20th international Conference on Computational Linguistics*. Geneva, Switzerland.
- J.C. Wu and J.S. Chang. 2007. *Learning to Find English to Chinese Transliterations on the Web*. In Proc. of EMNLP-CoNLL-2007. pp.996-1004. Prague, Czech Republic.
- Y. Zhang, S. Vogel. 2005 *Competitive Grouping in Integrated Phrase Segmentation and Alignment Model*. in *Proceedings of ACL-05 Workshop on Building and Parallel Text*. Ann Arbor, MI.

# Soft Syntactic Constraints for Hierarchical Phrased-Based Translation

Yuval Marton and Philip Resnik

Department of Linguistics

and the Laboratory for Computational Linguistics and Information Processing (CLIP)

at the Institute for Advanced Computer Studies (UMIACS)

University of Maryland, College Park, MD 20742-7505, USA

{ymarton, resnik} @t umiacs.umd.edu

## Abstract

In adding syntax to statistical MT, there is a tradeoff between taking advantage of linguistic analysis, versus allowing the model to exploit linguistically unmotivated mappings learned from parallel training data. A number of previous efforts have tackled this tradeoff by starting with a commitment to linguistically motivated analyses and then finding appropriate ways to soften that commitment. We present an approach that explores the tradeoff from the other direction, starting with a context-free translation model learned directly from aligned parallel text, and then adding soft constituent-level constraints based on parses of the source language. We obtain substantial improvements in performance for translation from Chinese and Arabic to English.

## 1 Introduction

The statistical revolution in machine translation, beginning with (Brown et al., 1993) in the early 1990s, replaced an earlier era of detailed language analysis with automatic learning of shallow source-target mappings from large parallel corpora. Over the last several years, however, the pendulum has begun to swing back in the other direction, with researchers exploring a variety of statistical models that take advantage of source- and particularly target-language syntactic analysis (e.g. (Cowan et al., 2006; Zollmann and Venugopal, 2006; Marcu et al., 2006; Galley et al., 2006) and numerous others).

Chiang (2005) distinguishes statistical MT approaches that are “syntactic” in a *formal* sense, go-

ing beyond the finite-state underpinnings of phrase-based models, from approaches that are syntactic in a *linguistic* sense, i.e. taking advantage of *a priori* language knowledge in the form of annotations derived from human linguistic analysis or treebanking.<sup>1</sup> The two forms of syntactic modeling are doubly dissociable: current research frameworks include systems that are finite state but informed by linguistic annotation prior to training (e.g., (Koehn and Hoang, 2007; Birch et al., 2007; Hassan et al., 2007)), and also include systems employing context-free models trained on parallel text without benefit of any prior linguistic analysis (e.g. (Chiang, 2005; Chiang, 2007; Wu, 1997)). Over time, however, there has been increasing movement in the direction of systems that are syntactic in both the formal and linguistic senses.

In any such system, there is a natural tension between taking advantage of the linguistic analysis, versus allowing the model to use linguistically unmotivated mappings learned from parallel training data. The tradeoff often involves starting with a system that exploits rich linguistic representations and relaxing some part of it. For example, DeNeefe et al. (2007) begin with a tree-to-string model, using treebank-based target language analysis, and find it useful to modify it in order to accommodate useful “phrasal” chunks that are present in parallel training data but not licensed by linguistically motivated parses of the target language. Similarly, Cowan et al. (2006) focus on using syntactically rich representations of source and target parse trees, but they resort to phrase-based translation for modifiers within

<sup>1</sup>See (Lopez, to appear) for a comprehensive survey.

clauses. Finding the right way to balance linguistic analysis with unconstrained data-driven modeling is clearly a key challenge.

In this paper we address this challenge from a less explored direction. Rather than starting with a system based on linguistically motivated parse trees, we begin with a model that is syntactic only in the formal sense. We then introduce soft constraints that take source-language parses into account to a limited extent. Introducing syntactic constraints in this restricted way allows us to take maximal advantage of what can be learned from parallel training data, while effectively factoring in key aspects of linguistically motivated analysis. As a result, we obtain substantial improvements in performance for both Chinese-English and Arabic-English translation.

In Section 2, we briefly review the Hiero statistical MT framework (Chiang, 2005, 2007), upon which this work builds, and we discuss Chiang’s initial effort to incorporate soft source-language constituency constraints for Chinese-English translation. In Section 3, we suggest that an insufficiently fine-grained view of constituency constraints was responsible for Chiang’s lack of strong results, and introduce finer grained constraints into the model. Section 4 demonstrates the value of these constraints via substantial improvements in Chinese-English translation performance, and extends the approach to Arabic-English. Section 5 discusses the results, and Section 6 considers related work. Finally we conclude in Section 7 with a summary and potential directions for future work.

## 2 Hierarchical Phrase-based Translation

### 2.1 Hiero

Hiero (Chiang, 2005; Chiang, 2007) is a hierarchical phrase-based statistical MT framework that generalizes phrase-based models by permitting phrases with gaps. Formally, Hiero’s translation model is a weighted synchronous context-free grammar. Hiero employs a generalization of the standard non-hierarchical phrase extraction approach in order to acquire the synchronous rules of the grammar directly from word-aligned parallel text. Rules have the form  $X \rightarrow \langle \bar{e}, \bar{f} \rangle$ , where  $\bar{e}$  and  $\bar{f}$  are phrases containing terminal symbols (words) and possibly co-indexed instances of the

nonterminal symbol  $X$ .<sup>2</sup> Associated with each rule is a set of translation model features,  $\phi_i(\bar{f}, \bar{e})$ ; for example, one intuitively natural feature of a rule is the phrase translation (log-)probability  $\phi(\bar{f}, \bar{e}) = \log p(\bar{e}|\bar{f})$ , directly analogous to the corresponding feature in non-hierarchical phrase-based models like Pharaoh (Koehn et al., 2003). In addition to this phrase translation probability feature, Hiero’s feature set includes the inverse phrase translation probability  $\log p(\bar{f}|\bar{e})$ , lexical weights  $\text{lexwt}(\bar{f}|\bar{e})$  and  $\text{lexwt}(\bar{e}|\bar{f})$ , which are estimates of translation quality based on word-level correspondences (Koehn et al., 2003), and a rule penalty allowing the model to learn a preference for longer or shorter derivations; see (Chiang, 2007) for details.

These features are combined using a log-linear model, with each synchronous rule contributing

$$\sum_i \lambda_i \phi_i(\bar{f}, \bar{e}) \quad (1)$$

to the total log-probability of a derived hypothesis. Each  $\lambda_i$  is a weight associated with feature  $\phi_i$ , and these weights are typically optimized using minimum error rate training (Och, 2003).

### 2.2 Soft Syntactic Constraints

When looking at Hiero rules, which are acquired automatically by the model from parallel text, it is easy to find many cases that seem to respect linguistically motivated boundaries. For example,

$$X \rightarrow \langle \text{jingtian } X_1, X_1 \text{ this year} \rangle,$$

seems to capture the use of *jingtian/this year* as a temporal modifier when building linguistic constituents such as noun phrases (*the election this year*) or verb phrases (*voted in the primary this year*). However, it is important to observe that nothing in the Hiero framework actually *requires* nonterminal symbols to cover linguistically sensible constituents, and in practice they frequently do not.<sup>3</sup>

<sup>2</sup>This is slightly simplified: Chiang’s original formulation of Hiero, which we use, has two nonterminal symbols,  $X$  and  $S$ . The latter is used only in two special “glue” rules that permit complete trees to be constructed via concatenation of subtrees when there is no better way to combine them.

<sup>3</sup>For example, this rule could just as well be applied with  $X_1$  covering the “phrase” *submitted and* to produce non-constituent substring *submitted and this year* in a hypothesis like *The budget was submitted and this year cuts are likely*.



Chiang (2005) conjectured that there might be value in allowing the Hiero model to favor hypotheses for which the synchronous derivation respects linguistically motivated source-language constituency boundaries, as identified using a parser. He tested this conjecture by adding a soft constraint in the form of a “constituency feature”: if a synchronous rule  $X \rightarrow \langle \bar{e}, \bar{f} \rangle$  is used in a derivation, and the span of  $\bar{f}$  is a constituent in the source-language parse, then a term  $\lambda_c$  is added to the model score in expression (1).<sup>4</sup> Unlike a hard constraint, which would simply prevent the application of rules violating syntactic boundaries, using the feature to introduce a *soft* constraint allows the model to boost the “goodness” for a rule if it is consistent with the source language constituency analysis, and to leave its score unchanged otherwise. The weight  $\lambda_c$ , like all other  $\lambda_i$ , is set via minimum error rate training, and that optimization process determines empirically the extent to which the constituency feature should be trusted.

Figure 1 illustrates the way the constituency feature worked, treating English as the source language for the sake of readability. In this example,  $\lambda_c$  would be added to the hypothesis score for any rule used in the hypothesis whose source side spanned *the minister, a speech, yesterday, gave a speech yesterday, or the minister gave a speech yesterday*. A rule translating, say, *minister gave a* as a unit would receive no such boost.

Chiang tested the constituency feature for Chinese-English translation, and obtained no significant improvement on the test set. The idea then seems essentially to have been abandoned; it does not appear in later discussions (Chiang, 2007).

### 3 Soft Syntactic Constraints, Revisited

On the face of it, there are any number of possible reasons Chiang’s (2005) soft constraint did not work – including, for example, practical issues like the quality of the Chinese parses.<sup>5</sup> However, we focus here on two conceptual issues underlying his use of source language syntactic constituents.

<sup>4</sup>Formally,  $\phi_c(\bar{f}, \bar{e})$  is defined as a binary feature, with value 1 if  $\bar{f}$  spans a source constituent and 0 otherwise. In the latter case  $\lambda_c \phi_c(\bar{f}, \bar{e}) = 0$  and the score in expression (1) is unaffected.

<sup>5</sup>In fact, this turns out not to be the issue; see Section 4.

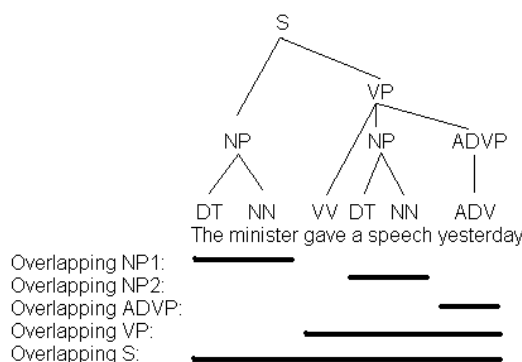


Figure 1: Illustration of Chiang’s (2005) syntactic constituency feature, which does not distinguish among constituent types.

First, the constituency feature treats all syntactic constituent types equally, making no distinction among them. For any given language pair, however, there might be some source constituents that tend to map naturally to the target language as units, and others that do not (Fox, 2002; Eisner, 2003). Moreover, a parser may tend to be more accurate for some constituents than for others.

Second, the Chiang (2005) constituency feature gives a rule additional credit when the rule’s source side overlaps exactly with a source-side syntactic constituent. Logically, however, it might make sense not just to give a rule  $X \rightarrow \langle \bar{e}, \bar{f} \rangle$  extra credit when  $\bar{f}$  matches a constituent, but to incur a *cost* when  $\bar{f}$  violates a constituent boundary. Using the example in Figure 1, we might want to penalize hypotheses containing rules where  $\bar{f}$  is *the minister gave a* (and other cases, such as *minister gave, minister gave a, and so forth*).<sup>6</sup>

These observations suggest a finer-grained approach to the constituency feature idea, retaining the idea of soft constraints, but applying them using *various* soft-constraint constituency features. Our first observation argues for distinguishing among constituent types (NP, VP, etc.). Our second observation argues for distinguishing the benefit of match-

<sup>6</sup>This accomplishes coverage of the logically complete set of possibilities, which include not only  $\bar{f}$  matching a constituent exactly or crossing its boundaries, but also  $\bar{f}$  being properly contained within the constituent span, properly containing it, or being outside it entirely. Whenever these latter possibilities occur,  $\bar{f}$  will exactly match or cross the boundaries of some other constituent.

ing constituents from the cost of crossing constituent boundaries. We therefore define a space of new features as the cross product

$$\{\text{CP, IP, NP, VP, } \dots\} \times \{=, +\}.$$

where = and + signify matching and crossing boundaries, respectively. For example,  $\phi_{\text{NP}=\}$  would denote a binary feature that matches whenever the span of  $\bar{f}$  exactly covers an NP in the source-side parse tree, resulting in  $\lambda_{\text{NP}=\}$  being added to the hypothesis score (expression (1)). Similarly,  $\phi_{\text{VP}+}$  would denote a binary feature that matches whenever the span of  $\bar{f}$  crosses a VP boundary in the parse tree, resulting in  $\lambda_{\text{VP}+}$  being *subtracted from* the hypothesis score.<sup>7</sup> For readability from this point forward, we will omit  $\phi$  from the notation and refer to features such as NP= (which one could read as “NP match”), VP+ (which one could read as “VP crossing”), etc.

In addition to these individual features, we define three more variants:

- For each constituent type, e.g. NP, we define a feature NP\_ that ties the weights of NP= and NP+. If NP= matches a rule, the model score is incremented by  $\lambda_{\text{NP}_=}$ , and if NP+ matches, the model score is decremented by the same quantity.
- For each constituent type, e.g. NP, we define a version of the model, NP2, in which NP= and NP+ are *both* included as features, with separate weights  $\lambda_{\text{NP}=\}$  and  $\lambda_{\text{NP}+}$ .
- We define a set of “standard” linguistic labels containing {CP, IP, NP, VP, PP, ADJP, ADVP, QP, LCP, DNP} and excluding other labels such as PRN (parentheses), FRAG (fragment), etc.<sup>8</sup> We define feature XP= as the disjunction of {CP=, IP=, ..., DNP=}; i.e. its value equals 1 for a rule if the span of  $\bar{f}$  exactly covers a constituent having any of the standard labels. The

<sup>7</sup>Formally,  $\lambda_{\text{VP}+}$  simply contributes to the sum in expression (1), as with all features in the model, but weight optimization using minimum error rate training should, and does, automatically assign this feature a negative weight.

<sup>8</sup>We map SBAR and S labels in Arabic parses to CP and IP, respectively, consistent with the Chinese parses. We map Chinese DP labels to NP. DNP and LCP appear only in Chinese. We ran no ADJP experiment in Chinese, because this label virtually always spans only one token in the Chinese parses.

definitions of XP+, XP\_, and XP2 are analogous.

- Similarly, since Chiang’s original constituency feature can be viewed as a disjunctive “all-labels=” feature, we also defined “all-labels+”, “all-labels2”, and “all-labels\_” analogously.

## 4 Experiments

We carried out MT experiments for translation from Chinese to English and from Arabic to English, using a descendant of Chiang’s Hiero system. Language models were built using the SRI Language Modeling Toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998). Word-level alignments were obtained using GIZA++ (Och and Ney, 2000). The baseline model in both languages used the feature set described in Section 2; for the Chinese baseline we also included a rule-based number translation feature (Chiang, 2007).

In order to compute syntactic features, we analyzed source sentences using state of the art, tree-bank trained constituency parsers ((Huang et al., 2008) for Chinese, and the Stanford parser v.2007-08-19 for Arabic (Klein and Manning, 2003a; Klein and Manning, 2003b)). In addition to the baseline condition, and baseline plus Chiang’s (2005) original constituency feature, experimental conditions augmented the baseline with additional features as described in Section 3.

All models were optimized and tested using the BLEU metric (Papineni et al., 2002) with the NIST-implemented (“shortest”) effective reference length, on lowercased, tokenized outputs/references. Statistical significance of difference from the baseline BLEU score was measured by using paired bootstrap re-sampling (Koehn, 2004).<sup>9</sup>

### 4.1 Chinese-English

For the Chinese-English translation experiments, we trained the translation model on the corpora in Table 1, totalling approximately 2.1 million sentence pairs after GIZA++ filtering for length ratio. Chinese text was segmented using the Stanford segmenter (Tseng et al., 2005).

<sup>9</sup>Whenever we use the word “significant”, we mean “statistically significant” (at  $p < .05$  unless specified otherwise).

LDC ID	Description
LDC2002E18	Xinhua Ch/Eng Par News V1 beta
LDC2003E07	Ch/En Treebank Par Corpus
LDC2005T10	Ch/En News Mag Par Txt (Sinorama)
LDC2003E14	FBIS Multilanguage Txts
LDC2005T06	Ch News Translation Txt Pt 1
LDC2004T08	HK Par Text (only HKNews)

Table 1: Training corpora for Chinese-English translation

We trained a 5-gram language model using the English (target) side of the training set, pruning 4-gram and 5-gram singletons. For minimum error rate training and development we used the NIST MTeval MT03 set.

Table 2 presents our results. We first evaluated translation performance using the NIST MT06 (nist-text) set. Like Chiang (2005), we find that the original, undifferentiated constituency feature (Chiang-05) introduces a negligible, statistically insignificant improvement over the baseline. However, we find that several of the finer-grained constraints (IP=, VP=, VP+, QP+, and NP=) achieve statistically significant improvements over baseline (up to .74 BLEU), and the latter three also improve significantly on the undifferentiated constituency feature. By combining multiple finer-grained syntactic features, we obtain significant improvements of up to 1.65 BLEU points (NP\_, VP2, IP2, all-labels\_, and XP+).

We also obtained further gains using combinations of features that had performed well; e.g., condition IP2.VP2.NP\_ augments the baseline features with IP2 and VP2 (i.e. IP=, IP+, VP= and VP+), and NP\_ (tying weights of NP= and NP+; see Section 3). Since component features in those combinations were informed by individual-feature performance on the test set, we tested the best performing conditions from MT06 on a new test set, NIST MT08. NP= and VP+ yielded significant improvements of up to 1.53 BLEU. Combination conditions replicated the pattern of results from MT06, including the same increasing order of gains, with improvements up to 1.11 BLEU.

## 4.2 Arabic-English

For Arabic-English translation, we used the training corpora in Table 3, approximately 100,000 sen-

Chinese	MT06	MT08
Baseline	.2624	.2064
Chiang-05	.2634	.2065
PP=	.2607	
DNP+	.2621	
CP+	.2622	
AP+	.2633	
AP=	.2634	
DNP=	.2640	
IP+	.2643	
PP+	.2644	
LCP=	.2649	
LCP+	.2654	
CP=	.2657	
NP+	.2662	
QP=	.2674 <sup>+</sup>	.2071
IP=	<b>.2680</b> <sup>*+</sup>	.2061
VP=	.2683 <sup>*</sup>	.2072
VP+	<b>.2693</b> <sup>**++</sup>	<b>.2109</b> <sup>*+</sup>
QP+	<b>.2694</b> <sup>**++</sup>	.2091
NP=	<b>.2698</b> <sup>**++</sup>	<b>.2217</b> <sup>**++</sup>
<b>Multiple / conflated features:</b>		
QP2	.2614	
NP2	.2621	
XP=	.2630	
XP2	.2633	
all-labels+	.2633	
VP_	.2637	
QP_	.2641	
NP.VP.IP=.QP.VP+	.2646	
IP_	.2647	
IP2+VP2	.2649	
all-labels2	.2673 <sup>*-</sup>	.2070
NP_	<b>.2690</b> <sup>**++</sup>	.2101 <sup>^+</sup>
IP2.VP2.NP_	<b>.2699</b> <sup>**++</sup>	<b>.2105</b> <sup>*+</sup>
VP2	<b>.2722</b> <sup>**++</sup>	<b>.2123</b> <sup>**++</sup>
all-labels_	<b>.2731</b> <sup>**++</sup>	<b>.2125</b> <sup>**++</sup>
IP2	<b>.2750</b> <sup>**++</sup>	<b>.2132</b> <sup>**+</sup>
XP+	<b>.2789</b> <sup>**++</sup>	<b>.2175</b> <sup>**++</sup>

Table 2: Chinese-English results. \*: Significantly better than baseline ( $p < .05$ ). \*\*: Significantly better than baseline ( $p < .01$ ). ^: Almost significantly better than baseline ( $p < .075$ ). +: Significantly better than Chiang-05 ( $p < .05$ ). ++: Significantly better than Chiang-05 ( $p < .01$ ). -: Almost significantly better than Chiang-05 ( $p < .075$ ).

LDC ID	Description
LDC2004T17	Ar News Trans Txt Pt 1
LDC2004T18	Ar/En Par News Pt 1
LDC2005E46	Ar/En Treebank En Translation
LDC2004E72	eTIRR Ar/En News Txt

Table 3: Training corpora for Arabic-English translation

tence pairs after GIZA++ length-ratio filtering. We trained a trigram language model using the English side of this training set, plus the English Gigaword v2 AFP and Gigaword v1 Xinhua corpora. Development and minimum error rate training were done using the NIST MT02 set.

Table 4 presents our results. We first tested on the NIST MT03 and MT06 (nist-text) sets. On MT03, the original, undifferentiated constituency feature did not improve over baseline. Two individual finer-grained features (PP+ and AdvP=) yielded statistically significant gains up to .42 BLEU points, and feature combinations AP2, XP2 and all-labels2 yielded significant gains up to 1.03 BLEU points. XP2 and all-labels2 also improved significantly on the undifferentiated constituency feature, by .72 and 1.11 BLEU points, respectively.

For MT06, Chiang’s original feature improved the baseline significantly — this is a new result using his feature, since he did not experiment with Arabic — as did our our IP=, PP=, and VP= conditions. Adding individual features PP+ and AdvP= yielded significant improvements up to 1.4 BLEU points over baseline, and in fact the improvement for individual feature AdvP= over Chiang’s undifferentiated constituency feature approaches significance ( $p < .075$ ).

More important, several conditions combining features achieved statistically significant improvements over baseline of up 1.94 BLEU points: XP2, IP2, IP, VP=.PP+.AdvP=, AP2, PP+.AdvP=, and AdvP2. Of these, AdvP2 is also a significant improvement over the undifferentiated constituency feature (Chiang-05), with  $p < .01$ . As we did for Chinese, we tested the best-performing models on a new test set, NIST MT08. Consistent patterns reappeared: improvements over the baseline up to 1.69 BLEU ( $p < .01$ ), with AdvP2 again in the lead (also outperforming the undifferentiated constituency feature,  $p < .05$ ).

<u>Arabic</u>	<u>MT03</u>	<u>MT06</u>	<u>MT08</u>
Baseline	.4795	.3571	.3571
<b>Chiang-05</b>	.4787	<b>.3679**</b>	<b>.3678**</b>
VP+	.4802	.3481	
AP+	.4856	.3495	
IP+	.4818	.3516	
CP=	.4815	.3523	
NP=	.4847	.3537	
NP+	.4800	.3548	
AP=	.4797	.3569	
AdvP+	.4852	.3572	
CP+	.4758	.3578	
<b>IP=</b>	.4811	<b>.3636**</b>	<b>.3647**</b>
<b>PP=</b>	.4801	<b>.3651**</b>	<b>.3662**</b>
<b>VP=</b>	.4803	<b>.3655**</b>	<b>.3694**</b>
<b>PP+</b>	<b>.4837**</b>	<b>.3707**</b>	<b>.3700**</b>
<b>AdvP=</b>	<b>.4823**</b>	<b>.3711**</b>	<b>.3717**</b>
<b>Multiple / conflated features:</b>			
XP+	.4771	.3522	
<b>all-labels2</b>	<b>.4898**+</b>	.3536	.3572
all-labels_	.4828	.3548	
VP2	.4826	.3552	
NP2	.4832	.3561	
AdvP.VP.PP.IP=	.4826	.3571	
VP_	.4825	.3604	
all-labels+	.4825	.3600	
<b>XP2</b>	<b>.4859**+</b>	.3605^	<b>.3613**</b>
<b>IP2</b>	.4793	<b>.3611*</b>	.3593
<b>IP_</b>	.4791	<b>.3635*</b>	<b>.3648**</b>
<b>XP=</b>	.4808	<b>.3659**</b>	<b>.3704**+</b>
VP=.PP+.AdvP=	<b>.4833**</b>	<b>.3677**</b>	<b>.3718**</b>
<b>AP2</b>	<b>.4840**</b>	<b>.3692**</b>	<b>.3719**</b>
<b>PP+.AdvP=</b>	.4777	<b>.3708**</b>	<b>.3680**</b>
<b>AdvP2</b>	.4803	<b>.3765**++</b>	<b>.3740**+</b>

Table 4: Arabic-English Experiments. Results are sorted by MT06 BLEU score. \*: Better than baseline ( $p < .05$ ). \*\*: Better than baseline ( $p < .01$ ). +: Better than Chiang-05 ( $p < .05$ ). ++: Better than Chiang-05 ( $p < .01$ ). -: Almost significantly better than Chiang-05 ( $p < .075$ )

## 5 Discussion

The results in Section 4 demonstrate, to our knowledge for the first time, that significant and sometimes substantial gains over baseline can be obtained by incorporating soft syntactic constraints into Hiero’s translation model. Within language, we also see considerable consistency across multiple test sets, in terms of which constraints tend to help most.

Furthermore, our results provide some insight into why the original approach may have failed to yield a positive outcome. For Chinese, we found that when we defined finer-grained versions of the exact-match features, there was value for some constituency types in biasing the model to favor matching the source language parse. Moreover, we found that there was significant value in allowing the model to be sensitive to violations (crossing boundaries) of source parses. These results confirm that parser quality was not the limitation in the original work (or at least not the only limitation), since in our experiments the parser was held constant.

Looking at combinations of new features, some “double-feature” combinations (VP2, IP2) achieved large gains, although note that more is not necessarily better: combinations of more features did not yield better scores, and some did not yield any gain at all. No conflated feature reached significance, but it is not the case that all conflated features are worse than their same-constituent “double-feature” counterparts. We found no simple correlation between finer-grained feature scores (and/or boundary condition type) and combination or conflation scores. Since some combinations seem to cancel individual contributions, we can conclude that the higher the number of participant features (of the kinds described here), the more likely a cancellation effect is; therefore, a “double-feature” combination is more likely to yield higher gains than a combination containing more features.

We also investigated whether non-canonical linguistic constituency labels such as PRN, FRAG, UCP and VSB introduce “noise”, by means of the XP features — the XP= feature is, in fact, simply the undifferentiated constituency feature, but sensitive only to “standard” XPs. Although performance of XP=, XP2 and all-labels+ were similar to that of the undifferentiated constituency feature, XP+ achieved

the highest gain. Intuitively, this seems plausible: the feature says, at least for Chinese, that a translation hypothesis should incur a penalty if it is translating a substring as a unit when that substring is not a canonical source constituent.

Having obtained positive results with Chinese, we explored the extent to which the approach might improve translation using a very different source language. The approach on Arabic-English translation yielded large BLEU gains over baseline, as well as significant improvements over the undifferentiated constituency feature. Comparing the two sets of experiments, we see that there are definitely language-specific variations in the value of syntactic constraints; for example, AdvP, the top performer in Arabic, cannot possibly perform well for Chinese, since in our parses the AdvP constituents rarely include more than a single word. At the same time, some IP and VP variants seem to do generally well in both languages. This makes sense, since — at least for these language pairs and perhaps more generally — clauses and verb phrases seem to correspond often on the source and target side. We found it more surprising that no NP variant yielded much gain in Arabic; this question will be taken up in future work.

## 6 Related Work

Space limitations preclude a thorough review of work attempting to navigate the tradeoff between using language analyzers and exploiting unconstrained data-driven modeling, although the recent literature is full of variety and promising approaches. We limit ourselves here to several approaches that seem most closely related.

Among approaches using parser-based syntactic models, several researchers have attempted to reduce the strictness of syntactic constraints in order to better exploit shallow correspondences in parallel training data. Our introduction has already briefly noted Cowan et al. (2006), who relax parse-tree-based alignment to permit alignment of non-constituent subphrases on the source side, and translate modifiers using a separate phrase-based model, and DeNeefe et al. (2007), who modify syntax-based extraction and binarize trees (following (Wang et al., 2007b)) to improve phrasal cov-

erage. Similarly, Marcu et al. (2006) relax their syntax-based system by rewriting target-side parse trees on the fly in order to avoid the loss of “non-syntactifiable” phrase pairs. Setiawan *et al.* (2007) employ a “function-word centered syntax-based approach”, with synchronous CFG and extended ITG models for reordering phrases, and relax syntactic constraints by only using a small number function words (approximated by high-frequency words) to guide the phrase-order inversion. Zollman and Venugopal (2006) start with a target language parser and use it to provide constraints on the extraction of hierarchical phrase pairs. Unlike Hiero, their translation model uses a full range of named nonterminal symbols in the synchronous grammar. As an alternative way to relax strict parser-based constituency requirements, they explore the use of phrases spanning generalized, categorial-style constituents in the parse tree, e.g. type NP/NN denotes a phrase like *the great* that lacks only a head noun (say, *wall*) in order to comprise an NP.

In addition, various researchers have explored the use of hard linguistic constraints on the source side, e.g. via “chunking” noun phrases and translating them separately (Owczarzak et al., 2006), or by performing hard reorderings of source parse trees in order to more closely approximate target-language word order (Wang et al., 2007a; Collins et al., 2005).

Finally, another soft-constraint approach that can also be viewed as coming from the data-driven side, adding syntax, is taken by Riezler and Maxwell (2006). They use LFG dependency trees on both source and target sides, and relax syntactic constraints by adding a “fragment grammar” for unparseable chunks. They decode using Pharaoh, augmented with their own log-linear features (such as  $p(e_{snippet}|f_{snippet})$  and its converse), side by side to “traditional” lexical weights. Riezler and Maxwell (2006) do not achieve higher BLEU scores, but do score better according to human grammaticality judgments for in-coverage cases.

## 7 Conclusion

When hierarchical phrase-based translation was introduced by Chiang (2005), it represented a new and successful way to incorporate syntax into statistical MT, allowing the model to exploit non-local depen-

dencies and lexically sensitive reordering without requiring linguistically motivated parsing of either the source or target language. An approach to incorporating parser-based constituents in the model was explored briefly, treating syntactic constituency as a soft constraint, with negative results.

In this paper, we returned to the idea of linguistically motivated soft constraints, and we demonstrated that they can, in fact, lead to substantial improvements in translation performance when integrated into the Hiero framework. We accomplished this using constraints that not only distinguish among constituent types, but which also distinguish between the benefit of matching the source parse bracketing, versus the cost of using phrases that cross relevant bracketing boundaries. We demonstrated improvements for Chinese-English translation, and succeed in obtaining substantial gains for Arabic-English translation, as well.

Our results contribute to a growing body of work on combining monolingually based, linguistically motivated syntactic analysis with translation models that are closely tied to observable parallel training data. Consistent with other researchers, we find that “syntactic constituency” may be too coarse a notion by itself; rather, there is value in taking a finer-grained approach, and in allowing the model to decide how far to trust each element of the syntactic analysis as part of the system’s optimization process.

## Acknowledgments

This work was supported in part by DARPA prime agreement HR0011-06-2-0001. The authors would like to thank David Chiang and Adam Lopez for making their source code available; the Stanford Parser team and Mary Harper for making their parsers available; David Chiang, Amy Weinberg, and CLIP Laboratory colleagues, particularly Chris Dyer, Adam Lopez, and Smaranda Muresan, for discussion and invaluable assistance.

## References

- Alexandra Birch, Miles Osborne, and Philipp Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation 2007*.
- P.F. Brown, S.A.D. Pietra, V.J.D. Pietra, and R.L. Mercer. 1993. The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–313.
- S. F. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Tech. Report TR-10-98, Comp. Sci. Group, Harvard U.
- David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL-05*, pages 263–270.
- David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
- Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL-05*.
- Brooke Cowan, Ivona Kucerova, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proc. EMNLP*.
- S DeNeefe, K. Knight, W. Wang, and D. Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proceedings of EMNLP-CoNLL*.
- J. Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *ACL Companion Vol.*
- Heidi Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proc. EMNLP 2002*.
- Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING/ACL-06*.
- H. Hassan, K. Sima'an, and A. Way. 2007. Integrating supertags into phrase-based statistical machine translation. In *Proc. ACL-07*, pages 288–295.
- Zhongqiang Huang, Denis Filimonov, and Mary Harper. 2008. Accuracy enhancements for mandarin parsing. Tech. report, University of Maryland.
- Dan Klein and Christopher D. Manning. 2003a. Accurate unlexicalized parsing. In *Proceedings of ACL-03*, pages 423–430.
- Dan Klein and Christopher D. Manning. 2003b. Fast exact inference with a factored model for natural language parsing. *Advances in Neural Information Processing Systems*, 15(NIPS 2002):3–10.
- Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proc. EMNLP+CoNLL*, pages 868–876, Prague.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133.
- Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*.
- Adam Lopez. (to appear). Statistical machine translation. *ACM Computing Surveys*. Earlier version: A Survey of Statistical Machine Translation. U. of Maryland, UMIACS tech. report 2006-47. Apr 2007.
- Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. EMNLP*, pages 44–52.
- Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the ACL*, pages 440–447. GIZA++.
- Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 160–167.
- K. Owczarzak, B. Mellebeek, D. Groves, J. Van Genabith, and A. Way. 2006. Wrapper syntax for example-based machine translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 148–155.
- Kishore Papineni, Salim Roukos, Todd Ward, John Henderson, and Florence Reeder. 2002. Corpusbased comprehensive and diagnostic MT evaluation: Initial Arabic, Chinese, French, and Spanish results. In *Proceedings of the Human Language Technology Conference (ACL'2002)*, pages 124–127, San Diego, CA.
- Stefan Riezler and John Maxwell. 2006. Grammatical machine translation. In *Proc. HLT-NAACL*, New York, NY.
- Hendra Setiawan, Min-Yen Kan, and Haizhou Li. 2007. Ordering phrases with function words. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 712–719.
- Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 901–904.
- Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *Fourth SIGHAN Workshop on Chinese Language Processing*.
- Chao Wang, Michael Collins, and Phillip Koehn. 2007a. Chinese syntactic reordering for statistical machine translation. In *Proceedings of EMNLP*.
- Wei Wang, Kevin Knight, and Daniel Marcu. 2007b. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proc. EMNLP+CoNLL 2007*.
- Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.
- Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the SMT Workshop, HLT-NAACL*.

# Generalizing Word Lattice Translation

Christopher Dyer\*, Smaranda Muresan, Philip Resnik\*

Laboratory for Computational Linguistics and Information Processing

Institute for Advanced Computer Studies

\*Department of Linguistics

University of Maryland

College Park, MD 20742, USA

redpony, smara, resnik AT umd.edu

## Abstract

Word lattice decoding has proven useful in spoken language translation; we argue that it provides a compelling model for translation of text genres, as well. We show that prior work in translating lattices using finite state techniques can be naturally extended to more expressive synchronous context-free grammar-based models. Additionally, we resolve a significant complication that non-linear word lattice inputs introduce in reordering models. Our experiments evaluating the approach demonstrate substantial gains for Chinese-English and Arabic-English translation.

## 1 Introduction

When Brown and colleagues introduced statistical machine translation in the early 1990s, their key insight – harkening back to Weaver in the late 1940s – was that translation could be viewed as an instance of noisy channel modeling (Brown et al., 1990). They introduced a now standard decomposition that distinguishes modeling sentences in the target language (language models) from modeling the relationship between source and target language (translation models). Today, virtually all statistical translation systems seek the best hypothesis  $e$  for a given input  $f$  in the source language, according to

$$\hat{e} = \arg \max_e Pr(e|f) \quad (1)$$

An exception is the translation of speech recognition output, where the acoustic signal generally underdetermines the choice of source word sequence  $f$ . There, Bertoldi and others have recently found that, rather than translating a single-best transcription  $f$ , it is advantageous to allow the MT decoder to

consider all possibilities for  $f$  by encoding the alternatives compactly as a confusion network or lattice (Bertoldi et al., 2007; Bertoldi and Federico, 2005; Koehn et al., 2007).

Why, however, should this advantage be limited to translation from spoken input? Even for text, there are often multiple ways to derive a sequence of words from the input string. Segmentation of Chinese, decomposing in German, morphological analysis for Arabic — across a wide range of source languages, ambiguity in the input gives rise to multiple possibilities for the source word sequence. Nonetheless, state-of-the-art systems commonly identify a single analysis  $f$  during a preprocessing step, and decode according to the decision rule in (1).

In this paper, we go beyond speech translation by showing that lattice decoding can also yield improvements for text by preserving alternative analyses of the input. In addition, we generalize lattice decoding algorithmically, extending it for the first time to hierarchical phrase-based translation (Chiang, 2005; Chiang, 2007).

Formally, the approach we take can be thought of as a “noisier channel”, where an observed signal  $o$  gives rise to a set of source-language strings  $f' \in \mathcal{F}(o)$  and we seek

$$\hat{e} = \arg \max_e \max_{f' \in \mathcal{F}(o)} Pr(e, f'|o) \quad (2)$$

$$= \arg \max_e \max_{f' \in \mathcal{F}(o)} Pr(e)Pr(f'|e, o) \quad (3)$$

$$= \arg \max_e \max_{f' \in \mathcal{F}(o)} Pr(e)Pr(f'|e)Pr(o|f') \quad (4)$$

Following Och and Ney (2002), we use the maximum entropy framework (Berger et al., 1996) to directly model the posterior  $Pr(e, f'|o)$  with parameters tuned to minimize a loss function representing



the quality only of the resulting translations. Thus, we make use of the following general decision rule:

$$\hat{e} = \arg \max_e \max_{f' \in \mathcal{F}(o)} \sum_{m=1}^M \lambda_m \phi_m(e, f', o) \quad (5)$$

In principle, one could decode according to (2) simply by enumerating and decoding each  $f' \in \mathcal{F}(o)$ ; however, for any interestingly large  $\mathcal{F}(o)$  this will be impractical. We assume that for many interesting cases of  $\mathcal{F}(o)$ , there will be identical substrings that express the same content, and therefore a lattice representation is appropriate.

In Section 2, we discuss decoding with this model in general, and then show how two classes of translation models can easily be adapted for lattice translation; we achieve a unified treatment of finite-state and hierarchical phrase-based models by treating lattices as a subcase of weighted finite state automata (FSAs). In Section 3, we identify and solve issues that arise with reordering in non-linear FSAs, i.e. FSAs where every path does not pass through every node. Section 4 presents two applications of the noisier channel paradigm, demonstrating substantial performance gains in Arabic-English and Chinese-English translation. In Section 5 we discuss relevant prior work, and we conclude in Section 6.

## 2 Decoding

Most statistical machine translation systems model translational equivalence using either finite state transducers or synchronous context free grammars (Lopez, to appear 2008). In this section we discuss the issues associated with adapting decoders from both classes of formalism to process word lattices. The first decoder we present is a SCFG-based decoder similar to the one described in Chiang (2007). The second is a phrase-based decoder implementing the model of Koehn et al. (2003).

### 2.1 Word lattices

A word lattice  $\mathcal{G} = \langle V, E \rangle$  is a directed acyclic graph that formally is a weighted finite state automaton (FSA). We further stipulate that exactly one node has no outgoing edges and is designated the ‘end node’. Figure 1 illustrates three classes of word lattices.

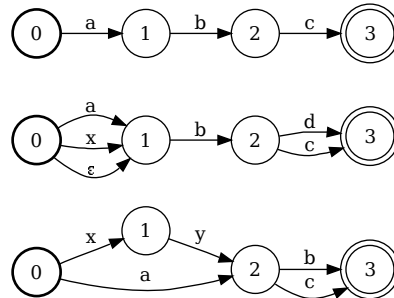


Figure 1: Three examples of word lattices: (a) sentence, (b) confusion network, and (c) non-linear word lattice.

A word lattice is useful for our purposes because it permits any finite set of strings to be represented and allows for substrings common to multiple members of the set to be represented with a single piece of structure. Additionally, all paths from one node to another form an equivalence class representing, in our model, alternative expressions of the same underlying communicative intent.

For translation, we will find it useful to encode  $\mathcal{G}$  in a chart based on a topological ordering of the nodes, as described by Cheppalier et al. (1999). The nodes in the lattices shown in Figure 1 are labeled according to an appropriate numbering.

The chart-representation of the graph is a triple of 2-dimensional matrices  $\langle \mathbf{F}, \mathbf{p}, \mathbf{R} \rangle$ , which can be constructed from the numbered graph.  $\mathbf{F}_{i,j}$  is the word label of the  $j^{\text{th}}$  transition leaving node  $i$ . The corresponding transition cost is  $\mathbf{p}_{i,j}$ .  $\mathbf{R}_{i,j}$  is the node number of the node on the *right* side of the  $j^{\text{th}}$  transition leaving node  $i$ . Note that  $\mathbf{R}_{i,j} > i$  for all  $i, j$ . Table 1 shows the word lattice from Figure 1 represented in matrix form as  $\langle \mathbf{F}, \mathbf{p}, \mathbf{R} \rangle$ .

	0	1	2
a	1 1	b 1 2	c 1 3
a	$\frac{1}{3}$ 1	b 1 2	c $\frac{1}{2}$ 3
x	$\frac{1}{3}$ 1		d $\frac{1}{2}$ 3
ε	$\frac{1}{3}$ 1		
x	$\frac{1}{2}$ 1	y 1 2	b $\frac{1}{2}$ 3
a	$\frac{1}{2}$ 2		c $\frac{1}{2}$ 3

Table 1: Topologically ordered chart encoding of the three lattices in Figure 1. Each cell  $ij$  in this table is a triple  $\langle \mathbf{F}_{ij}, \mathbf{p}_{ij}, \mathbf{R}_{ij} \rangle$

## 2.2 Parsing word lattices

Chiang (2005) introduced hierarchical phrase-based translation models, which are formally based on synchronous context-free grammars (SCFGs). Translation proceeds by parsing the input using the source language side of the grammar, simultaneously building a tree on the target language side via the target side of the synchronized rules. Since decoding is equivalent to parsing, we begin by presenting a parser for word lattices, which is a generalization of a CKY parser for lattices given in Cheppalier et al. (1999).

Following Goodman (1999), we present our lattice parser as a deductive proof system in Figure 2. The parser consists of two kinds of items, the first with the form  $[X \rightarrow \alpha \bullet \beta, i, j]$  representing rules that have yet to be completed and span node  $i$  to node  $j$ . The other items have the form  $[X, i, j]$  and indicate that non-terminal  $X$  spans  $[i, j]$ . As with sentence parsing, the goal is a deduction that covers the spans of the entire input lattice  $[S, 0, |V| - 1]$ .

The three inference rules are: 1) match a terminal symbol and move across one edge in the lattice 2) move across an  $\epsilon$ -edge without advancing the dot in an incomplete rule 3) advance the dot across a non-terminal symbol given appropriate antecedents.

## 2.3 From parsing to MT decoding

A target language model is necessary to generate fluent output. To do so, the grammar is intersected with an  $n$ -gram LM. To mitigate the effects of the combinatorial explosion of non-terminals the LM intersection entails, we use *cube-pruning* to only consider the most promising expansions (Chiang, 2007).

## 2.4 Lattice translation with FSTs

A second important class of translation models includes those based formally on FSTs. We present a description of the decoding process for a word lattice using a representative FST model, the phrase-based translation model described in Koehn et al. (2003).

Phrase-based models translate a foreign sentence  $f$  into the target language  $e$  by breaking up  $f$  into a sequence of phrases  $\bar{f}_1^f$ , where each phrase  $\bar{f}_i$  can contain one or more contiguous words and is translated into a target phrase  $e_i$  of one or more contiguous words. Each word in  $f$  must be translated ex-

actly once. To generalize this model to word lattices, it is necessary to choose both a path through the lattice and a partitioning of the sentence this induces into a sequence of phrases  $\bar{f}_1^f$ . Although the number of source phrases in a word lattice can be exponential in the number of nodes, enumerating the *possible translations* of every span in a lattice is in practice tractable, as described by Bertoldi et al. (2007).

## 2.5 Decoding with phrase-based models

We adapted the Moses phrase-based decoder to translate word lattices (Koehn et al., 2007). The unmodified decoder builds a translation hypothesis from left to right by selecting a range of untranslated words and adding translations of this phrase to the end of the hypothesis being extended. When no untranslated words remain, the translation process is complete.

The word lattice decoder works similarly, only now the decoder keeps track not of the words that have been covered, but of the *nodes*, given a topological ordering of the nodes. For example, assuming the third lattice in Figure 1 is our input, if the edge with word  $a$  is translated, this will cover *two* untranslated nodes  $[0,1]$  in the coverage vector, even though it is only a single word. As with sentence-based decoding, a translation hypothesis is complete when all nodes in the input lattice are covered.

## 2.6 Non-monotonicity and unreachable nodes

The changes described thus far are straightforward adaptations of the underlying phrase-based sentence decoder; however, dealing properly with non-monotonic decoding of word lattices introduces some minor complexity that is worth mentioning. In the sentence decoder, any translation of any span of untranslated words is an allowable extension of a partial translation hypothesis, provided that the coverage vectors of the extension and the partial hypothesis do not intersect. In a non-linear word lattice, a further constraint must be enforced ensuring that there is always a path from the starting node of the translation extension's source to the node representing the nearest right edge of the already-translated material, as well as a path from the ending node of the translation extension's source to future translated spans. Figure 3 illustrates the problem. If  $[0,1]$  is translated, the decoder must not consider translating

Axioms:

$$\overline{[X \rightarrow \bullet \gamma, i, i] : w} \quad (X \xrightarrow{w} \langle \gamma, \alpha \rangle) \in G, i \in [0, |V| - 2]$$

Inference rules:

$$\frac{[X \rightarrow \alpha \bullet \mathbf{F}_{j,k} \beta, i, j] : w}{[X \rightarrow \alpha \mathbf{F}_{j,k} \bullet \beta, i, \mathbf{R}_{j,k}] : w \times \mathbf{p}_{j,k}}$$

$$\frac{[X \rightarrow \alpha \bullet \beta, i, j] : w}{[X \rightarrow \alpha \bullet \beta, i, \mathbf{R}_{j,k}] : w \times \mathbf{p}_{j,k}} \quad \mathbf{F}_{j,k} = \epsilon$$

$$\frac{[Z \rightarrow \alpha \bullet X \beta, i, k] : w_1 \quad [X \rightarrow \gamma \bullet, k, j] : w_2}{[Z \rightarrow \alpha X \bullet \beta, i, j] : w_1 \times w_2}$$

Goal state:

$$[S \rightarrow \gamma \bullet, 0, |V| - 1]$$

Figure 2: Word lattice parser for an unrestricted context free grammar  $G$ .

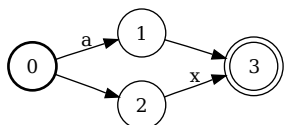


Figure 3: The span  $[0, 3]$  has one inconsistent covering,  $[0, 1] + [2, 3]$ .

$[2,3]$  as a possible extension of this hypothesis since there is no path from node 1 to node 2 and therefore the span  $[1,2]$  would never be covered. In the parser that forms the basis of the hierarchical decoder described in Section 2.3, no such restriction is necessary since grammar rules are processed in a strictly left-to-right fashion without any skips.

### 3 Distortion in a non-linear word lattice

In both hierarchical and phrase-based models, the distance between words in the source sentence is used to limit where in the target sequence their translations will be generated. In phrase based translation, distortion is modeled explicitly. Models that support non-monotonic decoding generally include a distortion cost, such as  $|a_i - b_{i-1} - 1|$  where  $a_i$  is the starting position of the foreign phrase  $\bar{f}_i$  and  $b_{i-1}$  is the ending position of phrase  $\bar{f}_{i-1}$  (Koehn et al., 2003). The intuition behind this model is that since most translation is monotonic, the cost of skipping ahead or back in the source should be proportional to the number of words that are skipped. Additionally, a maximum distortion limit is used to restrict

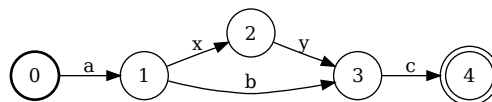


Figure 4: Distance-based distortion problem. What is the distance between node 4 to node 0?

the size of the search space.

In linear word lattices, such as confusion networks, the distance metric used for the distortion penalty and for distortion limits is well defined; however, in a non-linear word lattice, it poses the problem illustrated in Figure 4. Assuming the left-to-right decoding strategy described in the previous section, if  $c$  is generated by the first target word, the distortion penalty associated with “skipping ahead” should be either 3 or 2, depending on what path is chosen to translate the span  $[0,3]$ . In large lattices, where a single arc may span many nodes, the possible distances may vary quite substantially depending on what path is ultimately taken, and handling this properly therefore crucial.

Although hierarchical phrase-based models do not model distortion explicitly, Chiang (2007) suggests using a span length limit to restrict the window in which reordering can take place.<sup>1</sup> The decoder enforces the constraint that a synchronous rule learned from the training data (the only mechanism by which reordering can be introduced) can span

<sup>1</sup>This is done to reduce the size of the search space and because hierarchical phrase-based translation models are inaccurate models of long-distance distortion.

Distance metric	MT05	MT06
Difference	0.2943	0.2786
Difference+LexRO	0.2974	0.2890
ShortestP	0.2993	0.2865
ShortestP+LexRO	0.3072	0.2992

Table 2: Effect of distance metric on phrase-based model performance.

maximally  $\Lambda$  words in  $f$ . Like the distortion cost used in phrase-based systems,  $\Lambda$  is also poorly defined for non-linear lattices.

Since we want a distance metric that will restrict as few local reorderings as possible on *any* path, we use a function  $\xi(a, b)$  returning the length of the shortest path between nodes  $a$  and  $b$ . Since this function is not dependent on the exact path chosen, it can be computed in advance of decoding using an all-pairs shortest path algorithm (Cormen et al., 1989).

### 3.1 Experimental results

We tested the effect of the distance metric on translation quality using Chinese word segmentation lattices (Section 4.1, below) using both a hierarchical and phrase-based system modified to translate word lattices. We compared the shortest-path distance metric with a baseline which uses the difference in node number as the distortion distance. For an additional datapoint, we added a lexicalized reordering model that models the probability of each phrase pair appearing in three different orientations (swap, monotone, other) in the training corpus (Koehn et al., 2005).

Table 2 summarizes the results of the phrase-based systems. On both test sets, the shortest path metric improved the BLEU scores. As expected, the lexicalized reordering model improved translation quality over the baseline; however, the improvement was more substantial in the model that used the shortest-path distance metric (which was already a higher baseline). Table 3 summarizes the results of our experiment comparing the performance of two distance metrics to determine whether a rule has exceeded the decoder’s span limit. The pattern is the same, showing a clear increase in BLEU for the shortest path metric over the baseline.

Distance metric	MT05	MT06
Difference	0.3063	0.2957
ShortestP	0.3176	0.3043

Table 3: Effect of distance metric on hierarchical model performance.

## 4 Exploiting Source Language Alternatives

**Chinese word segmentation.** A necessary first step in translating Chinese using standard models is segmenting the character stream into a sequence of words. Word-lattice translation offers two possible improvements over the conventional approach. First, a lattice may represent multiple alternative segmentations of a sentence; input represented in this way will be more robust to errors made by the segmenter.<sup>2</sup> Second, different segmentation granularities may be more or less optimal for translating different spans. By encoding alternatives in the input in a word lattice, the decision as to which granularity to use for a given span can be resolved during decoding rather than when constructing the system. Figure 5 illustrates a lattice based on three different segmentations.

**Arabic morphological variation.** Arabic orthography is problematic for lexical and phrase-based MT approaches since a large class of functional elements (prepositions, pronouns, tense markers, conjunctions, definiteness markers) are attached to their host stems. Thus, while the training data may provide good evidence for the translation of a particular stem by itself, the same stem may not be attested when attached to a particular conjunction. The general solution taken is to take the best possible morphological analysis of the text (it is often ambiguous whether a piece of a word is part of the stem or merely a neighboring functional element), and then make a subset of the bound functional elements in the language into freestanding tokens. Figure 6 illustrates the unsegmented Arabic surface form as well as the morphological segmentation variant we made use of. The limitation of this approach is that as the amount and variety of training data increases, the optimal segmentation strategy changes: more aggressive segmentation results

<sup>2</sup>The segmentation process is ambiguous, even for native speakers of Chinese.

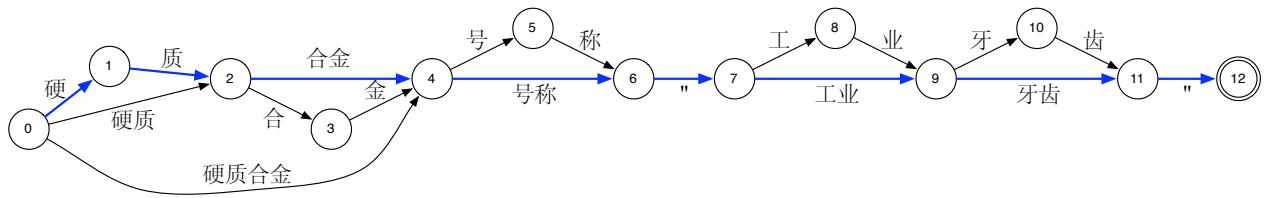


Figure 5: Sample Chinese segmentation lattice using three segmentations.

in fewer OOV tokens, but automatic evaluation metrics indicate lower translation quality, presumably because the smaller units are being translated less idiomatically (Habash and Sadat, 2006). Lattices allow the decoder to make decisions about what granularity of segmentation to use *subsentially*.

#### 4.1 Chinese Word Segmentation Experiments

In our experiments we used two state-of-the-art Chinese word segmenters: one developed at Harbin Institute of Technology (Zhao et al., 2001), and one developed at Stanford University (Tseng et al., 2005). In addition, we used a character-based segmentation. In the remaining of this paper, we use **CS** for character segmentation, **HS** for Harbin segmentation and **SS** for Stanford segmentation. We built two types of lattices: one that combines the Harbin and Stanford segmenters (**HS+SS**), and one which uses all three segmentations (**HS+SS+CS**).

**Data and Settings.** The systems used in these experiments were trained on the NIST MT06 Eval corpus without the UN data (approximately 950K sentences). The corpus was analyzed with the three segmentation schemes. For the systems using word lattices, the training data contained the versions of the corpus appropriate for the segmentation schemes used in the input. That is, for the **HS+SS** condition, the training data consisted of two copies of the corpus: one segmented with the Harbin segmenter and the other with the Stanford segmenter.<sup>3</sup> A trigram English language model with modified Kneser-Ney smoothing (Kneser and Ney, 1995) was trained on the English side of our training data as well as portions of the Gigaword v2 English Corpus, and was used for all experiments. The NIST MT03 test set was used as a development set for optimizing the interpolation weights using minimum error rate train-

<sup>3</sup>The corpora were word-aligned independently and then concatenated for rule extraction.

ing (Och, 2003). The testing was done on the NIST 2005 and 2006 evaluation sets (MT05, MT06).

**Experimental results: Word-lattices improve translation quality.** We used both a phrase-based translation model, decoded using our modified version of Moses (Koehn et al., 2007), and a hierarchical phrase-based translation model, using our modified version of Hiero (Chiang, 2005; Chiang, 2007). These two translation model types illustrate the applicability of the theoretical contributions presented in Section 2 and Section 3.

We observed that the coverage of named entities (NEs) in our baseline systems was rather poor. Since names in Chinese can be composed of relatively long strings of characters that cannot be translated individually, when generating the segmentation lattices that included **CS** arcs, we avoided segmenting NEs of type **PERSON**, as identified using a Chinese NE tagger (Florian et al., 2004).

The results are summarized in Table 4. We see that using word lattices improves BLEU scores both in the phrase-based model and hierarchical model as compared to the single-best segmentation approach. All results using our word-lattice decoding for the hierarchical models (**HS+SS** and **HS+SS+CS**) are significantly better than the best segmentation (**SS**).<sup>4</sup> For the phrase-based model, we obtain significant gains using our word-lattice decoder using all three segmentations on MT05. The other results, while better than the best segmentation (**HS**) by at least 0.3 BLEU points, are not statistically significant. Even if the results are not statistically significant for MT06, there is a high decrease in OOV items when using word-lattices. For example, for MT06 the number of OOVs in the **HS** translation is 484.

<sup>4</sup>Significance testing was carried out using the bootstrap resampling technique advocated by Koehn (2004). Unless otherwise noted, all reported improvements are significant at least  $p < 0.05$ .

surface	wx Al ftrp AlSyf kAn mEZm AlDjyj AlAE Amy m&y dA l EmAd .
segmented	w- x Al ftrp Al- Syf kAn mEZm Al- Djyj Al- AE Amy m&y dA l- Al- EmAd .
(English)	During the summer period , most media buzz was supportive of the general .

Figure 6: Example of Arabic morphological segmentation.

The number of OOVs decreased by 19% for **hs+ss** and by 75% for **hs+ss+cs**. As mentioned in Section 3, using lexical reordering for word-lattices further improves the translation quality.

## 4.2 Arabic Morphology Experiments

We created lattices from an unsegmented version of the Arabic test data and generated alternative arcs where clitics as well as the definiteness marker and the future tense marker were segmented into tokens. We used the Buckwalter morphological analyzer and disambiguated the analysis using a simple unigram model trained on the Penn Arabic Treebank.

**Data and Settings.** For these experiments we made use of the entire NIST MT08 training data, although for training of the system, we used a sub-sampling method proposed by Kishore Papineni that aims to include training sentences containing  $n$ -grams in the test data (personal communication). For all systems, we used a 5-gram English LM trained on 250M words of English training data. The NIST MT03 test set was used as development set for optimizing the interpolation weights using MER training (Och, 2003). Evaluation was carried out on the NIST 2005 and 2006 evaluation sets (MT05, MT06).

**Experimental results: Word-lattices improve translation quality.** Results are presented in Table 5. Using word-lattices to combine the surface forms with morphologically segmented forms significantly improves BLEU scores both in the phrase-based and hierarchical models.

## 5 Prior work

**Lattice Translation.** The ‘noisier channel’ model of machine translation has been widely used in spoken language translation as an alternative to selecting the single-best hypothesis from an ASR system and translating it (Ney, 1999; Casacuberta et al., 2004; Zhang et al., 2005; Saleem et al., 2005; Matusov et al., 2005; Bertoldi et al., 2007; Mathias, 2007). Several authors (e.g. Saleem et al. (2005)

and Bertoldi et al. (2007)) comment directly on the impracticality of using  $n$ -best lists to translate speech.

Although translation is fundamentally a non-monotonic relationship between most language pairs, reordering has tended to be a secondary concern to the researchers who have worked on lattice translation. Matusov et al. (2005) decodes monotonically and then uses a finite state reordering model on the single-best translation, along the lines of Bangalore and Riccardi (2000). Mathias (2007) and Saleem et al. (2004) only report results of monotonic decoding for the systems they describe. Bertoldi et al. (2007) solve the problem by requiring that their input be in the format of a confusion network, which enables the standard distortion penalty to be used. Finally, the system described by Zhang et al. (2005) uses IBM Model 4 features to translate lattices. For the distortion model, they use the maximum probability value over all possible paths in the lattice for each jump considered, which is similar to the approach we have taken. Mathias and Byrne (2006) build a phrase-based translation system as a cascaded series of FSTs which can accept any input FSA; however, the only reordering that is permitted is the swapping of two adjacent phrases.

Applications of source lattices outside of the domain of spoken language translation have been far more limited. Costa-jussà and Fonollosa (2007) take steps in this direction by using lattices to encode multiple reorderings of the source language. Dyer (2007) uses confusion networks to encode morphological alternatives in Czech-English translation, and Xu et al. (2005) takes an approach very similar to ours for Chinese-English translation and encodes multiple word segmentations in a lattice, but which is decoded with a conventionally trained translation model and without a sophisticated reordering model.

The Arabic-English morphological segmentation lattices are similar in spirit to backoff translation models (Yang and Kirchhoff, 2006), which consider alternative morphological segmentations and simpli-

(Source Type)	MT05 BLEU	MT06 BLEU
cs	0.2833	0.2694
hs	0.2905	0.2835
ss	0.2894	0.2801
hs+ss	0.2938	0.2870
hs+ss+cs	0.2993	0.2865
hs+ss+cs.lexRo	0.3072	0.2992

(a) Phrase-based model

(Source Type)	MT05 BLEU	MT06 BLEU
cs	0.2904	0.2821
hs	0.3008	0.2907
ss	0.3071	0.2964
hs+ss	0.3132	0.3006
hs+ss+cs	0.3176	0.3043

(b) Hierarchical model

Table 4: Chinese Word Segmentation Results

(Source Type)	MT05 BLEU	MT06 BLEU
surface	0.4682	0.3512
morph	0.5087	0.3841
morph+surface	0.5225	0.4008

(a) Phrase-based model

(Source Type)	MT05 BLEU	MT06 BLEU
surface	0.5253	0.3991
morph	0.5377	0.4180
morph+surface	0.5453	0.4287

(b) Hierarchical model

Table 5: Arabic Morphology Results

fications of a surface token when the surface token can not be translated.

**Parsing and formal language theory.** There has been considerable work on parsing word lattices, much of it for language modeling applications in speech recognition (Ney, 1991; Cheppalier and Rajman, 1998). Additionally, Grune and Jacobs (2008) refines an algorithm originally due to Bar-Hillel for intersecting an arbitrary FSA (of which word lattices are a subset) with a CFG. Klein and Manning (2001) formalize parsing as a hypergraph search problem and derive an  $O(n^3)$  parser for lattices.

## 6 Conclusions

We have achieved substantial gains in translation performance by decoding compact representations of alternative source language analyses, rather than single-best representations. Our results generalize previous gains for lattice translation of spoken language input, and we have further generalized the approach by introducing an algorithm for lattice decoding using a hierarchical phrase-based model. Additionally, we have shown that although word lattices complicate modeling of word reordering, a simple heuristic offers good performance and enables many standard distortion models to be used directly with lattice input.

## Acknowledgments

This research was supported by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-0001. The authors wish to thank Niyu Ge for the Chinese named-entity analysis, Pi-Chuan Chang for her assistance with the Stanford Chinese segmenter, and Tie-Jun Zhao and Congui Zhu for making the Harbin Chinese segmenter available to us.

## References

- S. Bangalore and G. Riccardi. 2000. Finite state models for lexical reordering in spoken language translation. In *Proc. Int. Conf. on Spoken Language Processing*, pages 422–425, Beijing, China.
- A.L. Berger, V.J. Della Pietra, and S.A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.
- N. Bertoldi and M. Federico. 2005. A new decoder for spoken language translation based on confusion networks. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*.
- N. Bertoldi, R. Zens, and M. Federico. 2007. Speech translation by confusion network decoding. In *Proceeding of ICASSP 2007*, Honolulu, Hawaii, April.
- P.F. Brown, J. Cocke, S. Della-Pietra, V.J. Della-Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, and P.S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16:79–85.
- F. Casacuberta, H. Ney, F. J. Och, E. Vidal, J. M. Vilar, S. Barrachina, I. Garcia-Varea, D. Llorens, C. Mar-

- tinez, S. Molau, F. Nevado, M. Pastor, D. Pico, A. San-chis, and C. Tillmann. 2004. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech & Language*, 18(1):25–47, January.
- J. Cheppalier and M. Rajman. 1998. A generalized CYK algorithm for parsing stochastic CFG. In *Proceedings of the Workshop on Tabulation in Parsing and Deduction (TAPD98)*, pages 133–137, Paris, France.
- J. Cheppalier, M. Rajman, R. Aragues, and A. Rozenknop. 1999. Lattice parsing for speech recognition. In *Sixth Conference sur le Traitement Automatique du Langage Naturel (TANL'99)*, pages 95–104.
- D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270.
- D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
- T.H. Cormen, C. E. Leiserson, and R. L. Rivest, 1989. *Introduction to Algorithms*, pages 558–565. The MIT Press and McGraw-Hill Book Company.
- M. Costa-jussà and J.A.R. Fonollosa. 2007. Analysis of statistical and morphological classes to generate weighted reordering hypotheses on a statistical machine translation system. In *Proc. of the Second Workshop on SMT*, pages 171–176, Prague.
- C. Dyer. 2007. Noisier channel translation: translation from morphologically complex languages. In *Proceedings of the Second Workshop on Statistical Machine Translation*, Prague, June.
- R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. of HLT-NAACL 2004*, pages 1–8.
- J. Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25:573–605.
- D. Grune and C.J. H. Jacobs. 2008. Parsing as intersection. *Parsing Techniques*, pages 425–442.
- N. Habash and F. Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proc. of NAACL*, New York.
- D. Klein and C. D. Manning. 2001. Parsing with hypergraphs. In *Proceedings of IWPT 2001*.
- R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.
- P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL 2003*, pages 48–54.
- P. Koehn, A. Axelrod, A. Birch Mayne, C. Callison-Burch, M. Osborne, and D. Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proc. of IWSLT 2005*, Pittsburgh.
- P. Koehn, H. Hoang, A. Birch Mayne, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL), Demonstration Session*, pages 177–180, Jun.
- P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of the 2004 Conf. on EMNLP*, pages 388–395.
- A. Lopez. to appear 2008. Statistical machine translation. *ACM Computing Surveys*.
- L. Mathias and W. Byrne. 2006. Statistical phrase-based speech translation. In *IEEE Conf. on Acoustics, Speech and Signal Processing*.
- L. Mathias. 2007. *Statistical Machine Translation and Automatic Speech Recognition under Uncertainty*. Ph.D. thesis, The Johns Hopkins University.
- E. Matusov, S. Kanthak, and H. Ney. 2005. On the integration of speech recognition and statistical machine translation. In *Proceedings of Interspeech 2005*.
- H. Ney. 1991. Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing*, 39(2).
- H. Ney. 1999. Speech translation: Coupling of recognition and translation. In *Proc. of ICASSP*, pages 517–520, Phoenix.
- F. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 295–302.
- S. Saleem, S.-C. Jou, S. Vogel, and T. Schulz. 2005. Using word lattice information for a tighter coupling in speech translation systems. In *Proc. of ICSLP*, Jeju Island, Korea.
- H. Tseng, P. Chang, G. Andrew, D. Jurafsky, and C. Manning. 2005. A conditional random field word segmenter. In *Fourth SIGHAN Workshop on Chinese Language Processing*.
- J. Xu, E. Matusov, R. Zens, and H. Ney. 2005. Integrated Chinese word segmentation in statistical machine translation. In *Proc. of IWSLT 2005*, Pittsburgh.
- M. Yang and K. Kirchhoff. 2006. Phrase-based back-off models for machine translation of highly inflected languages. In *Proceedings of the EACL 2006*, pages 41–48.
- R. Zhang, G. Kikui, H. Yamamoto, and W. Lo. 2005. A decoding algorithm for word lattice translation in speech translation. In *Proceedings of the 2005 International Workshop on Spoken Language Translation*.
- T. Zhao, L. Yajuan, Y. Muyun, and Y. Hao. 2001. Increasing accuracy of chinese segmentation with strategy of multi-step processing. In *J Chinese Information Processing (Chinese Version)*, volume 1, pages 13–18.



# Combining Multiple Resources to Improve SMT-based Paraphrasing Model\*

Shiqi Zhao<sup>1</sup>, Cheng Niu<sup>2</sup>, Ming Zhou<sup>2</sup>, Ting Liu<sup>1</sup>, Sheng Li<sup>1</sup>

<sup>1</sup>Harbin Institute of Technology, Harbin, China

{zhaosq, tliu, lisheng}@ir.hit.edu.cn

<sup>2</sup>Microsoft Research Asia, Beijing, China

{chengniu, mingzhou}@microsoft.com

## Abstract

This paper proposes a novel method that exploits multiple resources to improve statistical machine translation (SMT) based paraphrasing. In detail, a phrasal paraphrase table and a feature function are derived from each resource, which are then combined in a log-linear SMT model for sentence-level paraphrase generation. Experimental results show that the SMT-based paraphrasing model can be enhanced using multiple resources. The phrase-level and sentence-level precision of the generated paraphrases are above 60% and 55%, respectively. In addition, the contribution of each resource is evaluated, which indicates that all the exploited resources are useful for generating paraphrases of high quality.

## 1 Introduction

Paraphrases are alternative ways of conveying the same meaning. Paraphrases are important in many natural language processing (NLP) applications, such as machine translation (MT), question answering (QA), information extraction (IE), multi-document summarization (MDS), and natural language generation (NLG).

This paper addresses the problem of sentence-level paraphrase generation, which aims at generating paraphrases for input sentences. An example of sentence-level paraphrases can be seen below:

*S1: The table was set up in the carriage shed.*

*S2: The table was laid under the cart-shed.*

---

\*This research was finished while the first author worked as an intern in Microsoft Research Asia.

Paraphrase generation can be viewed as monolingual machine translation (Quirk et al., 2004), which typically includes a translation model and a language model. The translation model can be trained using monolingual parallel corpora. However, acquiring such corpora is not easy. Hence, data sparseness is a key problem for the SMT-based paraphrasing. On the other hand, various methods have been presented to extract phrasal paraphrases from different resources, which include thesauri, monolingual corpora, bilingual corpora, and the web. However, little work has been focused on using the extracted phrasal paraphrases in sentence-level paraphrase generation.

In this paper, we exploit multiple resources to improve the SMT-based paraphrase generation. In detail, six kinds of resources are utilized, including: (1) an automatically constructed thesaurus, (2) a monolingual parallel corpus from novels, (3) a monolingual comparable corpus from news articles, (4) a bilingual phrase table, (5) word definitions from Encarta dictionary, and (6) a corpus of similar user queries. Among the resources, (1), (2), (3), and (4) have been investigated by other researchers, while (5) and (6) are first used in this paper. From those resources, six phrasal paraphrase tables are extracted, which are then used in a log-linear SMT-based paraphrasing model.

Both phrase-level and sentence-level evaluations were carried out in the experiments. In the former one, phrase substitutes occurring in the paraphrase sentences were evaluated. While in the latter one, the acceptability of the paraphrase sentences was evaluated. Experimental results show that: (1) The

SMT-based paraphrasing is enhanced using multiple resources. The phrase-level and sentence-level precision of the generated paraphrases exceed 60% and 55%, respectively. (2) Although the contributions of the resources differ a lot, all the resources are useful. (3) The performance of the method varies greatly on different test sets and it performs best on the test set of news sentences, which are from the same source as most of the training data.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 introduces the log-linear model for paraphrase generation. Section 4 describes the phrasal paraphrase extraction from different resources. Section 5 presents the parameter estimation method. Section 6 shows the experiments and results. Section 7 draws the conclusion.

## 2 Related Work

Paraphrases have been used in many NLP applications. In MT, Callison-Burch et al. (2006) utilized paraphrases of unseen source phrases to alleviate data sparseness. Kauchak and Barzilay (2006) used paraphrases of the reference translations to improve automatic MT evaluation. In QA, Lin and Pantel (2001) and Ravichandran and Hovy (2002) paraphrased the answer patterns to enhance the recall of answer extraction. In IE, Shinyama et al. (2002) automatically learned paraphrases of IE patterns to reduce the cost of creating IE patterns by hand. In MDS, McKeown et al. (2002) identified paraphrase sentences across documents before generating summarizations. In NLG, Iordanskaja et al. (1991) used paraphrases to generate more varied and fluent texts.

Previous work has examined various resources for acquiring paraphrases, including thesauri, monolingual corpora, bilingual corpora, and the web. Thesauri, such as WordNet, have been widely used for extracting paraphrases. Some researchers extract synonyms as paraphrases (Kauchak and Barzilay, 2006), while some others use looser definitions, such as hypernyms and holonyms (Barzilay and Elhadad, 1997). Besides, the automatically constructed thesauri can also be used. Lin (1998) constructed a thesaurus by automatically clustering words based on context similarity.

Barzilay and McKeown (2001) used monolingual parallel corpora for identifying paraphrases. They

exploited a corpus of multiple English translations of the same source text written in a foreign language, from which phrases in aligned sentences that appear in similar contexts were extracted as paraphrases. In addition, Finch et al. (2005) applied MT evaluation methods (BLEU, NIST, WER and PER) to build classifiers for paraphrase identification.

Monolingual parallel corpora are difficult to find, especially in non-literature domains. Alternatively, some researchers utilized monolingual comparable corpora for paraphrase extraction. Different news articles reporting on the same event are commonly used as monolingual comparable corpora, from which both paraphrase patterns and phrasal paraphrases can be derived (Shinyama et al., 2002; Barzilay and Lee, 2003; Quirk et al., 2004).

Lin and Pantel (2001) learned paraphrases from a parsed monolingual corpus based on an extended distributional hypothesis, where if two paths in dependency trees tend to occur in similar contexts it is hypothesized that the meanings of the paths are similar. The monolingual corpus used in their work is not necessarily parallel or comparable. Thus it is easy to obtain. However, since this resource is used to extract paraphrase patterns other than phrasal paraphrases, we do not use it in this paper.

Bannard and Callison-Burch (2005) learned phrasal paraphrases using bilingual parallel corpora. The basic idea is that if two phrases are aligned to the same translation in a foreign language, they may be paraphrases. This method has been demonstrated effective in extracting large volume of phrasal paraphrases. Besides, Wu and Zhou (2003) exploited bilingual corpora and translation information in learning synonymous collocations.

In addition, some researchers extracted paraphrases from the web. For example, Ravichandran and Hovy (2002) retrieved paraphrase patterns from the web using hand-crafted queries. Pasca and Dienes (2005) extracted sentence fragments occurring in identical contexts as paraphrases from one billion web documents. Since web mining is rather time consuming, we do not exploit the web to extract paraphrases in this paper.

So far, two kinds of methods have been proposed for sentence-level paraphrase generation, i.e., the pattern-based and SMT-based methods. Automatically learned patterns have been used in para-

phrase generation. For example, Barzilay and Lee (2003) applied multiple-sequence alignment (MSA) to parallel news sentences and induced paraphrasing patterns for generating new sentences. Pang et al. (2003) built finite state automata (FSA) from semantically equivalent translation sets based on syntactic alignment and used the FSAs in paraphrase generation. The pattern-based methods can generate complex paraphrases that usually involve syntactic variation. However, the methods were demonstrated to be of limited generality (Quirk et al., 2004).

Quirk et al. (2004) first recast paraphrase generation as monolingual SMT. They generated paraphrases using a SMT system trained on parallel sentences extracted from clustered news articles. In addition, Madnani et al. (2007) also generated sentence-level paraphrases based on a SMT model. The advantage of the SMT-based method is that it achieves better coverage than the pattern-based method. The main difference between their methods and ours is that they only used bilingual parallel corpora as paraphrase resource, while we exploit and combine multiple resources.

### 3 SMT-based Paraphrasing Model

The SMT-based paraphrasing model used by Quirk et al. (2004) was the noisy channel model of Brown et al. (1993), which identified the optimal paraphrase  $T^*$  of a sentence  $S$  by finding:

$$\begin{aligned} T^* &= \arg \max_T \{P(T|S)\} \\ &= \arg \max_T \{P(S|T)P(T)\} \end{aligned} \quad (1)$$

In contrast, we adopt a log-linear model (Och and Ney, 2002) in this work, since multiple paraphrase tables can be easily combined in the log-linear model. Specifically, feature functions are derived from each paraphrase resource and then combined with the language model feature<sup>1</sup>:

$$T^* = \arg \max_T \left\{ \sum_{i=1}^N \lambda_{TM.i} h_{TM.i}(T, S) + \lambda_{LM} h_{LM}(T, S) \right\} \quad (2)$$

where  $N$  is the number of paraphrase tables.  $h_{TM.i}(T, S)$  is the feature function based on the  $i$ -th paraphrase table  $PT_i$ .  $h_{LM}(T, S)$  is the language

<sup>1</sup>The reordering model is not considered in our model.

model feature.  $\lambda_{TM.i}$  and  $\lambda_{LM}$  are the weights of the feature functions.  $h_{TM.i}(T, S)$  is defined as:

$$h_{TM.i}(T, S) = \log \prod_{k=1}^{K_i} \text{Score}_i(T_k, S_k) \quad (3)$$

where  $K_i$  is the number of phrase substitutes from  $S$  to  $T$  based on  $PT_i$ .  $T_k$  in  $T$  and  $S_k$  in  $S$  are phrasal paraphrases in  $PT_i$ .  $\text{Score}_i(T_k, S_k)$  is the paraphrase likelihood according to  $PT_i$ <sup>2</sup>. A 5-gram language model is used, therefore:

$$h_{LM}(T, S) = \log \prod_{j=1}^J p(t_j | t_{j-4}, \dots, t_{j-1}) \quad (4)$$

where  $J$  is the length of  $T$ ,  $t_j$  is the  $j$ -th word of  $T$ .

## 4 Exploiting Multiple Resources

This section describes the extraction of phrasal paraphrases using various resources. Similar to Pharaoh (Koehn, 2004), our decoder<sup>3</sup> uses top 20 paraphrase options for each input phrase in the default setting. Therefore, we keep at most 20 paraphrases for a phrase when extracting phrasal paraphrases using each resource.

**1 - Thesaurus:** The thesaurus<sup>4</sup> used in this work was automatically constructed by Lin (1998). The similarity of two words  $e_1$  and  $e_2$  was calculated through the surrounding context words that have dependency relations with the investigated words:

$$\begin{aligned} \text{Sim}(e_1, e_2) &= \frac{\sum_{(r,e) \in T_r(e_1) \cap T_r(e_2)} (I(e_1, r, e) + I(e_2, r, e))}{\sum_{(r,e) \in T_r(e_1)} I(e_1, r, e) + \sum_{(r,e) \in T_r(e_2)} I(e_2, r, e)} \end{aligned} \quad (5)$$

where  $T_r(e_i)$  denotes the set of words that have dependency relation  $r$  with word  $e_i$ .  $I(e_i, r, e)$  is the mutual information between  $e_i$ ,  $r$  and  $e$ .

For each word, we keep 20 most similar words as paraphrases. In this way, we extract 502,305 pairs of paraphrases. The paraphrasing score  $\text{Score}_1(p_1, p_2)$  used in Equation (3) is defined as the similarity based on Equation (5).

<sup>2</sup>If none of the phrase substitutes from  $S$  to  $T$  is from  $PT_i$  (i.e.,  $K_i = 0$ ), we cannot compute  $h_{TM.i}(T, S)$  as in Equation (3). In this case, we assign  $h_{TM.i}(T, S)$  a minimum value.

<sup>3</sup>The decoder used here is a re-implementation of Pharaoh.

<sup>4</sup><http://www.cs.ualberta.ca/~lindek/downloads.htm>.

**2 - Monolingual parallel corpus:** Following Barzilay and McKeown (2001), we exploit a corpus of multiple English translations of foreign novels, which contains 25,804 parallel sentence pairs. We find that most paraphrases extracted using the method of Barzilay and McKeown (2001) are quite short. Thus we employ a new approach for paraphrase extraction. Specifically, we parse the sentences with CollinsParser<sup>5</sup> and extract the chunks from the parsing results. Let  $S_1$  and  $S_2$  be a pair of parallel sentences,  $p_1$  and  $p_2$  two chunks from  $S_1$  and  $S_2$ , we compute the similarity of  $p_1$  and  $p_2$  as:

$$Sim(p_1, p_2) = \alpha Sim_{content}(p_1, p_2) + (1 - \alpha) Sim_{context}(p_1, p_2) \quad (6)$$

where,  $Sim_{content}(p_1, p_2)$  is the content similarity, which is the word overlapping rate of  $p_1$  and  $p_2$ .  $Sim_{context}(p_1, p_2)$  is the context similarity, which is the word overlapping rate of the contexts of  $p_1$  and  $p_2$ <sup>6</sup>. If the similarity of  $p_1$  and  $p_2$  exceeds a threshold  $Th_1$ , they are identified as paraphrases. We extract 18,698 pairs of phrasal paraphrases from this resource. The paraphrasing score  $Score_2(p_1, p_2)$  is defined as the similarity in Equation (6). For the paraphrases occurring more than once, we use their maximum similarity as the paraphrasing score.

**3 - Monolingual comparable corpus:** Similar to the methods in (Shinyama et al., 2002; Barzilay and Lee, 2003), we construct a corpus of comparable documents from a large corpus  $D$  of news articles. The corpus  $D$  contains 612,549 news articles. Given articles  $d_1$  and  $d_2$  from  $D$ , if their publication date interval is less than 2 days and their similarity<sup>7</sup> exceeds a threshold  $Th_2$ , they are recognized as comparable documents. In this way, a corpus containing 5,672,864 pairs of comparable documents is constructed. From the comparable corpus, parallel sentences are extracted. Let  $s_1$  and  $s_2$  be two sentences from comparable documents  $d_1$  and  $d_2$ , if their similarity based on word overlapping rate is above a threshold  $Th_3$ ,  $s_1$  and  $s_2$  are identified as parallel sentences. In this way, 872,330 parallel sentence pairs are extracted.

<sup>5</sup><http://people.csail.mit.edu/mcollins/code.html>

<sup>6</sup>The context of a chunk is made up of 6 words around the chunk, 3 to the left and 3 to the right.

<sup>7</sup>The similarity of two documents is computed using the vector space model and the word weights are based on tf-idf.

We run Giza++ (Och and Ney, 2000) on the parallel sentences and then extract aligned phrases as described in (Koehn, 2004). The generated paraphrase table is pruned by keeping the top 20 paraphrases for each phrase. After pruning, 100,621 pairs of paraphrases are extracted. Given phrase  $p_1$  and its paraphrase  $p_2$ , we compute  $Score_3(p_1, p_2)$  by relative frequency (Koehn et al., 2003):

$$Score_3(p_1, p_2) = p(p_2|p_1) = \frac{count(p_2, p_1)}{\sum_{p'} count(p', p_1)} \quad (7)$$

People may wonder why we do not use the same method on the monolingual parallel and comparable corpora. This is mainly because the volumes of the two corpora differ a lot. In detail, the monolingual parallel corpus is fairly small, thus automatic word alignment tool like Giza++ may not work well on it. In contrast, the monolingual comparable corpus is quite large, hence we cannot conduct the time-consuming syntactic parsing on it as we do on the monolingual parallel corpus.

**4 - Bilingual phrase table:** We first construct a bilingual phrase table that contains 15,352,469 phrase pairs from an English-Chinese parallel corpus. We extract paraphrases from the bilingual phrase table and compute the paraphrasing score of phrases  $p_1$  and  $p_2$  as in (Bannard and Callison-Burch, 2005):

$$Score_4(p_1, p_2) = \sum_f p(f|p_1)p(p_2|f) \quad (8)$$

where  $f$  denotes a Chinese translation of both  $p_1$  and  $p_2$ .  $p(f|p_1)$  and  $p(p_2|f)$  are the translation probabilities provided by the bilingual phrase table. For each phrase, the top 20 paraphrases are kept according to the score in Equation (8). As a result, 3,177,600 pairs of phrasal paraphrases are extracted.

**5 - Encarta dictionary definitions:** Words and their definitions can be regarded as paraphrases. Here are some examples from Encarta dictionary: “*hurricane: severe storm*”, “*clever: intelligent*”, “*travel: go on journey*”. In this work, we extract words’ definitions from Encarta dictionary web pages<sup>8</sup>. If a word has more than one definition, all of them are extracted. Note that the words and definitions in the

<sup>8</sup><http://encarta.msn.com/encnet/features/dictionary/dictionaryhome.aspx>

dictionary are lemmatized, but words in sentences are usually inflected. Hence, we expand the word - definition pairs by providing the inflected forms. Here we use an inflection list and some rules for inflection. After expanding, 159,456 pairs of phrasal paraphrases are extracted. Let  $\langle p_1, p_2 \rangle$  be a word - definition pair, the paraphrasing score is defined according to the rank of  $p_2$  in all of  $p_1$ 's definitions:

$$Score_5(p_1, p_2) = \gamma^{i-1} \quad (9)$$

where  $\gamma$  is a constant (we empirically set  $\gamma = 0.9$ ) and  $i$  is the rank of  $p_2$  in  $p_1$ 's definitions.

**6 - Similar user queries:** Clusters of similar user queries have been used for query expansion and suggestion (Gao et al., 2007). Since most queries are at the phrase level, we exploit similar user queries as phrasal paraphrases. In our experiment, we use the corpus of clustered similar MSN queries constructed by Gao et al. (2007). The similarity of two queries  $p_1$  and  $p_2$  is computed as:

$$Sim(p_1, p_2) = \beta Sim_{content}(p_1, p_2) + (1 - \beta) Sim_{click-through}(p_1, p_2) \quad (10)$$

where  $Sim_{content}(p_1, p_2)$  is the content similarity, which is computed as the word overlapping rate of  $p_1$  and  $p_2$ .  $Sim_{click-through}(p_1, p_2)$  is the click through similarity, which is the overlapping rate of the user clicked documents for  $p_1$  and  $p_2$ . For each query  $q$ , we keep the top 20 similar queries, whose similarity with  $q$  exceeds a threshold  $Th_4$ . As a result, 395,284 pairs of paraphrases are extracted. The score  $Score_6(p_1, p_2)$  is defined as the similarity in Equation (10).

**7 - Self-paraphrase:** In addition to the six resources introduced above, a special paraphrase table is used, which is made up of pairs of identical words. The reason why this paraphrase table is necessary is that a word should be allowed to keep unchanged in paraphrasing. This is a difference between paraphrasing and MT, since all words should be translated in MT. In our experiments, all the words that occur in the six paraphrase table extracted above are gathered to form the self-paraphrase table, which contains 110,403 word pairs. The score  $Score_7(p_1, p_2)$  is set 1 for each identical word pair.

## 5 Parameter Estimation

The weights of the feature functions, namely  $\lambda_{TM.i}$  ( $i = 1, 2, \dots, 7$ ) and  $\lambda_{LM}$ , need estimation<sup>9</sup>. In MT, the max-BLEU algorithm is widely used to estimate parameters. However, it may not work in our case, since it is more difficult to create a reference set of paraphrases.

We propose a new technique to estimate parameters in paraphrasing. The assumption is that, since a SMT-based paraphrase is generated through phrase substitution, we can measure the quality of a generated paraphrase by measuring its phrase substitutes. Generally, the paraphrases containing more correct phrase substitutes are judged as better paraphrases<sup>10</sup>. We therefore present the phrase substitution error rate (PSER) to score a generated paraphrase  $T$ :

$$PSER(T) = \|PS_0(T)\|/\|PS(T)\| \quad (11)$$

where  $PS(T)$  is the set of phrase substitutes in  $T$  and  $PS_0(T)$  is the set of incorrect substitutes.

In practice, we keep top  $n$  paraphrases for each sentence  $S$ . Thus we calculate the PSER for each source sentence  $S$  as:

$$PSER(S) = \|\bigcup_{i=1}^n PS_0(T_i)\|/\|\bigcup_{i=1}^n PS(T_i)\| \quad (12)$$

where  $T_i$  is the  $i$ -th generated paraphrase of  $S$ .

Suppose there are  $N$  sentences in the development set, the overall PSER is computed as:

$$PSER = \sum_{j=1}^N PSER(S_j) \quad (13)$$

where  $S_j$  is the  $j$ -th sentence in the development set.

Our development set contains 75 sentences (described in detail in Section 6). For each sentence, all possible phrase substitutes are extracted from the six paraphrase tables above. The extracted phrase substitutes are then manually labeled as ‘‘correct’’ or ‘‘incorrect’’. A phrase substitute is considered as correct only if the two phrases have the same meaning in the given sentence and the sentence generated by

<sup>9</sup>Note that, we also use some other parameters when extracting phrasal paraphrases from different resources, such as the thresholds  $Th_1, Th_2, Th_3, Th_4$ , as well as  $\alpha$  and  $\beta$  in Equation (6) and (10). These parameters are estimated using different development sets from the investigated resources. We do not describe the estimation of them due to space limitation.

<sup>10</sup>Paraphrasing a word to itself (based on the 7-th paraphrase table above) is not regarded as a substitute.

substituting the source phrase with the target phrase remains grammatical. In decoding, the phrase substitutes are printed out and then the PSER is computed based on the labeled data.

Using each set of parameters, we generate paraphrases for the sentences in the development set based on Equation (2). PSER is then computed as in Equation (13). We use the gradient descent algorithm (Press et al., 1992) to minimize PSER on the development set and get the optimal parameters.

## 6 Experiments

To evaluate the performance of the method on different types of test data, we used three kinds of sentences for testing, which were randomly extracted from Google news, free online novels, and forums, respectively. For each type, 50 sentences were extracted as test data and another 25 were extracted as development data. For each test sentence, top 10 of the generated paraphrases were kept for evaluation.

### 6.1 Phrase-level Evaluation

The phrase-level evaluation was carried out to investigate the contributions of the paraphrase tables. For each test sentence, all possible phrase substitutes were first extracted from the paraphrase tables and manually labeled as “correct” or “incorrect”. Here, the criterion for identifying paraphrases is the same as that described in Section 5. Then, in the stage of decoding, the phrase substitutes were printed out and evaluated using the labeled data.

Two metrics were used here. The first is the number of distinct correct substitutes (#DCS). Obviously, the more distinct correct phrase substitutes a paraphrase table can provide, the more valuable it is. The second is the accuracy of the phrase substitutes, which is computed as:

$$Accuracy = \frac{\#correct\ phrase\ substitutes}{\#all\ phrase\ substitutes} \quad (14)$$

To evaluate the PTs learned from different resources, we first used each PT (from 1 to 6) along with PT-7 in decoding. The results are shown in Table 1. It can be seen that PT-4 is the most useful, as it provides the most correct substitutes and the accuracy is the highest. We believe that it is because PT-4 is much larger than the other PTs. Compared with PT-4, the accuracies of the other PTs are fairly

PT combination	#DCS	Accuracy
1+7	178	14.61%
2+7	94	25.06%
3+7	202	18.35%
4+7	553	56.93%
5+7	231	20.48%
6+7	21	14.42%

Table 1: Contributions of the paraphrase tables.

PT-1: from the thesaurus; PT-2: from the monolingual parallel corpus; PT-3: from the monolingual comparable corpus; PT-4: from the bilingual parallel corpus; PT-5: from the Encarta dictionary definitions; PT-6: from the similar MSN user queries; PT-7: self-paraphrases.

low. This is because those PTs are smaller, thus they can provide fewer correct phrase substitutes. As a result, plenty of incorrect substitutes were included in the top 10 generated paraphrases.

PT-6 provides the least correct phrase substitutes and the accuracy is the lowest. There are several reasons. First, many phrases in PT-6 are not real phrases but only sets of keywords (e.g., “*lottery results ny*”), which may not appear in sentences. Second, many words in this table have spelling mistakes (e.g., “*widows vista*”). Third, some phrase pairs in PT-6 are not paraphrases but only “related queries” (e.g., “*back tattoo*” vs. “*butterfly tattoo*”). Fourth, many phrases of PT-6 contain proper names or out-of-vocabulary words, which are difficult to be matched. The accuracy based on PT-1 is also quite low. We found that it is mainly because the phrase pairs in PT-1 are automatically clustered, many of which are merely “similar” words rather than synonyms (e.g., “*borrow*” vs. “*buy*”).

Next, we try to find out whether it is necessary to combine all PTs. Thus we conducted several runs, each of which added the most useful PT from the left ones. The results are shown in Table 2. We can see that all the PTs are useful, as each PT provides some new correct phrase substitutes and the accuracy increases when adding each PT except PT-1.

Since the PTs are extracted from different resources, they have different contributions. Here we only discuss the contributions of PT-5 and PT-6, which are first used in paraphrasing in this paper. PT-5 is useful for paraphrasing uncommon concepts since it can “explain” concepts with their definitions.

PT combination	#DCS	Accuracy
4+7	553	56.93%
4+5+7	581	58.97%
4+5+3+7	638	59.42%
4+5+3+2+7	649	60.15%
4+5+3+2+1+7	699	60.14%
4+5+3+2+1+6+7	711	60.16%

Table 2: Performances of different combinations of paraphrase tables.

For instance, in the following test sentence  $S_1$ , the word “*amnesia*” is a relatively uncommon word, especially for the people using English as the second language. Based on PT-5,  $S_1$  can be paraphrased into  $T_1$ , which is much easier to understand.

$S_1$ : I was suffering from **amnesia**.  
 $T_1$ : I was suffering from **memory loss**.

The disadvantage of PT-5 is that substituting words with the definitions sometimes leads to grammatical errors. For instance, substituting “*heat shield*” in the sentence  $S_2$  with “*protective barrier against heat*” keeps the meaning unchanged. However, the paraphrased sentence  $T_2$  is ungrammatical.

$S_2$ : The U.S. space agency has been cautious about **heat shield** damage.  
 $T_2$ : The U.S. space administration has been cautious about **protective barrier against heat** damage.

As previously mentioned, PT-6 is less effective compared with the other PTs. However, it is useful for paraphrasing some special phrases, such as digital products, computer software, etc, since these phrases often appear in user queries. For example,  $S_3$  below can be paraphrased into  $T_3$  using PT-6.

$S_3$ : I have a **canon powershot** S230 that uses CF memory cards.  
 $T_3$ : I have a **canon digital camera** S230 that uses CF memory cards.

The phrase “*canon powershot*” can hardly be paraphrased using the other PTs. It suggests that PT-6 is useful for paraphrasing new emerging concepts and expressions.

Test sentences	Top-1	Top-5	Top-10
All 150	55.33%	45.20%	39.28%
50 from news	70.00%	62.00%	57.03%
50 from novel	56.00%	46.00%	37.42%
50 from forum	40.00%	27.60%	23.34%

Table 3: Top-n accuracy on different test sentences.

## 6.2 Sentence-level Evaluation

In this section, we evaluated the sentence-level quality of the generated paraphrases<sup>11</sup>. In detail, each generated paraphrase was manually labeled as “acceptable” or “unacceptable”. Here, the criterion for counting a sentence  $T$  as an acceptable paraphrase of sentence  $S$  is that  $T$  is understandable and its meaning is not evidently changed compared with  $S$ . For example, for the sentence  $S_4$ ,  $T_4$  is an acceptable paraphrase generated using our method.

$S_4$ : The **strain on US forces** of fighting in Iraq and Afghanistan **was exposed** yesterday when the Pentagon **published a report** showing that the **number** of suicides among US **troops** is at its **highest level** since the 1991 Gulf war.  
 $T_4$ : The **pressure on US troops** of fighting in Iraq and Afghanistan **was revealed** yesterday when the Pentagon **released a report** showing that the **amount** of suicides among US **forces** is at its **top** since the 1991 Gulf conflict.

We carried out sentence-level evaluation using the top-1, top-5, and top-10 results of each test sentence. The accuracy of the top-n results was computed as:

$$Accuracy_{top-n} = \frac{\sum_{i=1}^N n_i}{N \times n} \quad (15)$$

where  $N$  is the number of test sentences.  $n_i$  is the number of acceptable paraphrases in the top-n paraphrases of the  $i$ -th test sentence.

We computed the accuracy on the whole test set (150 sentences) as well as on the three subsets, i.e., the 50 news sentences, 50 novel sentences, and 50 forum sentences. The results are shown in table 3.

It can be seen that the accuracy varies greatly on different test sets. The accuracy on the news sentences is the highest, while that on the forum sentences is the lowest. There are several reasons. First,

<sup>11</sup>The evaluation was based on the paraphrasing results using the combination of all seven PTs.

the largest PT used in the experiments is extracted using the bilingual parallel data, which are mostly from news documents. Thus, the test set of news sentences is more similar to the training data.

Second, the news sentences are formal while the novel and forum sentences are less formal. Especially, some of the forum sentences contain spelling mistakes and grammar mistakes.

Third, we find in the results that, most phrases paraphrased in the novel and forum sentences are commonly used phrases or words, such as “*food*”, “*good*”, “*find*”, etc. These phrases are more difficult to paraphrase than the less common phrases, since they usually have much more paraphrases in the PTs. Therefore, it is more difficult to choose the right paraphrase from all the candidates when conducting sentence-level paraphrase generation.

Fourth, the forum sentences contain plenty of words such as “*board* (means *computer board*)”, “*site* (means *web site*)”, “*mouse* (means *computer mouse*)”, etc. These words are polysemous and have particular meanings in the domains of computer science and internet. Our method performs poor when paraphrasing these words since the domain of a context sentence is hard to identify.

After observing the results, we find that there are three types of errors: (1) syntactic errors: the generated sentences are ungrammatical. About 32% of the unacceptable results are due to syntactic errors. (2) semantic errors: the generated sentences are incomprehensible. Nearly 60% of the unacceptable paraphrases have semantic errors. (3) non-paraphrase: the generated sentences are well formed and comprehensible but are not paraphrases of the input sentences. 8% of the unacceptable results are of this type. We believe that many of the errors above can be avoided by applying syntactic constraints and by making better use of context information in decoding, which is left as our future work.

## 7 Conclusion

This paper proposes a method that improves the SMT-based sentence-level paraphrase generation using phrasal paraphrases automatically extracted from different resources. Our contribution is that we combine multiple resources in the framework of SMT for paraphrase generation, in which the dic-

tionary definitions and similar user queries are first used as phrasal paraphrases. In addition, we analyze and compare the contributions of different resources.

Experimental results indicate that although the contributions of the exploited resources differ a lot, they are all useful to sentence-level paraphrase generation. Especially, the dictionary definitions and similar user queries are effective for paraphrasing some certain types of phrases.

In the future work, we will try to use syntactic and context constraints in paraphrase generation to enhance the acceptability of the paraphrases. In addition, we will extract paraphrase patterns that contain more structural variation and try to combine the SMT-based and pattern-based systems for sentence-level paraphrase generation.

## Acknowledgments

We would like to thank Mu Li for providing us with the SMT decoder. We are also grateful to Dongdong Zhang for his help in the experiments.

## References

- Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, pages 597-604.
- Regina Barzilay and Michael Elhadad. 1997. Using Lexical Chains for Text Summarization. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 10-17.
- Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of HLT-NAACL*, pages 16-23.
- Regina Barzilay and Kathleen R. McKeown. 2001. Extracting Paraphrases from a Parallel Corpus. In *Proceedings of ACL*, pages 50-57.
- Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. In *Computational Linguistics* 19(2): 263-311.
- Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Translation Using Paraphrases. In *Proceedings of HLT-NAACL*, pages 17-24.
- Andrew Finch, Young-Sook Hwang, and Eiichiro Sumita. 2005. Using Machine Translation Evaluation Techniques to Determine Sentence-level Semantic Equivalence. In *Proceedings of IWP*, pages 17-24.



- Wei Gao, Cheng Niu, Jian-Yun Nie, Ming Zhou, Jian Hu, Kam-Fai Wong, and Hsiao-Wuen Hon. 2007. Cross-Lingual Query Suggestion Using Query Logs of Different Languages. In *Proceedings of SIGIR*, pages 463-470.
- Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. 1991. Lexical Selection and Paraphrase in a Meaning-Text Generation Model. In *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 293-312.
- David Kauchak and Regina Barzilay. 2006. Paraphrasing for Automatic Evaluation. In *Proceedings of HLT-NAACL*, pages 455-462.
- Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models: User Manual and Description for Version 1.2.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL*, pages 127-133.
- De-Kang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of COLING/ACL*, pages 768-774.
- De-Kang Lin and Patrick Pantel. 2001. Discovery of Inference Rules for Question Answering. In *Natural Language Engineering* 7(4): 343-360.
- Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie J. Dorr. 2007. Using Paraphrases for Parameter Tuning in Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 120-127.
- Kathleen R. Mckeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L. Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. 2002. Tracking and Summarizing News on a Daily Basis with Columbia's Newsblaster. In *Proceedings of HLT*, pages 280-285.
- Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of ACL*, pages 440-447.
- Franz Josef Och and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of ACL*, pages 295-302.
- Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based Alignment of Multiple Translations: Extracting Paraphrases and Generating New Sentences. In *Proceedings of HLT-NAACL*, pages 102-109.
- Marius Pasca and Péter Dienes. 2005. Aligning Needles in a Haystack: Paraphrase Acquisition Across the Web. In *Proceedings of IJCNLP*, pages 119-130.
- William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, U.K., 1992, 412-420.
- Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of EMNLP*, pages 142-149.
- Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of ACL*, pages 41-47.
- Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic Paraphrase Acquisition from News Articles. In *Proceedings of HLT*, pages 40-46.
- Hua Wu and Ming Zhou. 2003. Synonymous Collocation Extraction Using Translation Information. In *Proceedings of ACL*, pages 120-127.

# Extraction of Entailed Semantic Relations Through Syntax-based Comma Resolution

Vivek Srikumar<sup>1</sup> Roi Reichart<sup>2</sup> Mark Sammons<sup>1</sup> Ari Rappoport<sup>2</sup> Dan Roth<sup>1</sup>

<sup>1</sup>University of Illinois at Urbana-Champaign  
{vsrikum2|mssammon|danr}@uiuc.edu

<sup>2</sup>Institute of Computer Science, Hebrew University of Jerusalem  
{roiri|arir}@cs.huji.ac.il

## Abstract

This paper studies textual inference by investigating comma structures, which are highly frequent elements whose major role in the extraction of semantic relations has not been hitherto recognized. We introduce the problem of comma resolution, defined as understanding the role of commas and extracting the relations they imply. We show the importance of the problem using examples from Textual Entailment tasks, and present *A Sentence Transformation Rule Learner (ASTRL)*, a machine learning algorithm that uses a syntactic analysis of the sentence to learn sentence transformation rules that can then be used to extract relations. We have manually annotated a corpus identifying comma structures and relations they entail and experimented with both gold standard parses and parses created by a leading statistical parser, obtaining F-scores of 80.2% and 70.4% respectively.

## 1 Introduction

Recognizing relations expressed in text sentences is a major topic in NLP, fundamental in applications such as Textual Entailment (or Inference), Question Answering and Text Mining. In this paper we address this issue from a novel perspective, that of understanding the role of the commas in a sentence, which we argue is a key component in sentence comprehension. Consider for example the following three sentences:

1. *Authorities have arrested John Smith, a retired police officer.*
2. *Authorities have arrested John Smith, his friend and his brother.*

3. *Authorities have arrested John Smith, a retired police officer announced this morning.*

Sentence (1) states that John Smith is a retired police officer. The comma and surrounding sentence structure represent the relation ‘IsA’. In (2), the comma and surrounding structure signifies a list, so the sentence states that three people were arrested: (i) John Smith, (ii) his friend, and (iii) his brother. In (3), a retired police officer announced that John Smith has been arrested. Here, the comma and surrounding sentence structure indicate clause boundaries.

In all three sentences, the comma and the surrounding sentence structure signify relations essential to comprehending the meaning of the sentence, in a way that is not easily captured using lexical or even shallow parse-level information. As a human reader, we understand them easily, but automated systems for Information Retrieval, Question Answering, and Textual Entailment are likely to encounter problems when comparing structures like these, which are lexically similar, but whose meanings are so different.

In this paper we present an algorithm for *comma resolution*, a task that we define to consist of (1) disambiguating comma type and (2) determining the relations entailed from the sentence given the commas’ interpretation. Specifically, in (1) we assign each comma to one of five possible types, and in (2) we generate a set of natural language sentences that express the relations, if any, signified by each comma structure. The algorithm uses information extracted from parse trees. This work, in addition to having immediate significance for natural language processing systems that use semantic content, has potential applications in improving a range of auto-

mated analysis by decomposing complex sentences into a set of simpler sentences that capture the same meaning. Although there are many other widely-used structures that express relations in a similar way, commas are one of the most commonly used symbols<sup>1</sup>. By addressing comma resolution, we offer a promising first step toward resolving relations in sentences.

To evaluate the algorithm, we have developed annotation guidelines, and manually annotated sentences from the WSJ PennTreebank corpus. We present a range of experiments showing the good performance of the system, using gold-standard and parser-generated parse trees.

In Section 2 we motivate comma resolution through Textual Entailment examples. Section 3 describes related work. Sections 4 and 5 present our corpus annotation and learning algorithm. Results are given in Section 6.

## 2 Motivating Comma Resolution Through Textual Entailment

Comma resolution involves not only comma disambiguation but also inference of the arguments (and argument boundaries) of the relationship represented by the comma structure, and the relationships holding between these arguments and the sentence as a whole. To our knowledge, this is the first paper that deals with this problem, so in this section we motivate it in depth by showing its importance to the semantic inference task of Textual Entailment (TE) (Dagan et al., 2006), which is increasingly recognized as a crucial direction for improving a range of NLP tasks such as information extraction, question answering and summarization.

TE is the task of deciding whether the meaning of a text  $T$  (usually a short snippet) can be inferred from the meaning of another text  $S$ . If this is the case, we say that  $S$  entails  $T$ . For example<sup>2</sup>, we say that sentence (1) entails sentence (2):

1.  $S$ : *Parviz Davudi was representing Iran at a meeting of the Shanghai Co-operation Organization (SCO), the fledgling association that*

<sup>1</sup>For example, the WSJ corpus has 49K sentences, among which 32K with one comma or more, 17K with two or more, and 7K with three or more.

<sup>2</sup>The examples of this section are variations of pairs taken from the Pascal RTE3 (Dagan et al., 2006) dataset.

*binds two former Soviet republics of central Asia, Russia and China to fight terrorism.*

2.  $T$ : *SCO is the fledgling association that binds several countries.*

To see that (1) entails (2), one must understand that the first comma structure in sentence (1) is an apposition structure, and does not indicate the beginning of a list. The second comma marks a boundary between entities in a list. To make the correct inference one must determine that the second comma is a list separator, not an apposition marker. Misclassifying the second comma in (1) as an apposition leads to the conclusion that (1) entails (3):

3.  $T$ : *Russia and China are two former Soviet republics of central Asia .*

Note that even to an educated native speaker of English, sentence 1 may be initially confusing; during the first reading, one might interpret the first comma as indicating a list, and that ‘the Shanghai Co-operation Organization’ and ‘the fledgling association that binds...’ are two separate entities that are meeting, rather than two representations of the same entity.

From these examples we draw the following conclusions: 1. Comma resolution is essential in comprehending natural language text. 2. Explicitly representing relations derived from comma structures can assist a wide range of NLP tasks; this can be done by directly augmenting the lexical-level representation, e.g., by bringing surface forms of two text fragments with the same meaning closer together. 3. Comma structures might be highly ambiguous, nested and overlapping, and consequently their interpretation is a difficult task. The argument boundaries of the corresponding extracted relations are also not easy to detect.

The output of our system could be used to augment sentences with an explicit representation of entailed relations that hold in them. In Textual Entailment systems this can increase the likelihood of correct identification of entailed sentences, and in other NLP systems it can help understanding the shallow lexical/syntactic content of a sentence. A similar approach has been taken in (Bar-Haim et al., 2007; de Salvo Braz et al., 2005), which augment the source sentence with entailed relations.

### 3 Related Work

Since we focus on extracting the relations represented by commas, there are two main strands of research with similar goals: 1) systems that directly analyze commas, whether labeling them with syntactic information or correcting inappropriate use in text; and 2) systems that extract relations from text, typically by trying to identify paraphrases.

The significance of interpreting the role of commas in sentences has already been identified by (van Delden and Gomez, 2002; Bayraktar et al., 1998) and others. A review of the first line of research is given in (Say and Akman, 1997).

In (Bayraktar et al., 1998) the WSJ PennTreebank corpus (Marcus et al., 1993) is analyzed and a very detailed list of syntactic patterns that correspond to different roles of commas is created. However, they do not study the extraction of entailed relations as a function of the comma's interpretation. Furthermore, the syntactic patterns they identify are unlexicalized and would not support the level of semantic relations that we show in this paper. Finally, theirs is a manual process completely dependent on syntactic patterns. While our comma resolution system uses syntactic parse information as its main source of features, the approach we have developed focuses on the *entailed relations*, and does not limit implementations to using only syntactic information.

The most directly comparable prior work is that of (van Delden and Gomez, 2002), who use finite state automata and a greedy algorithm to learn comma syntactic roles. However, their approach differs from ours in a number of critical ways. First, their comma annotation scheme does not identify arguments of predicates, and therefore cannot be used to extract complete relations. Second, for each comma type they identify, a new Finite State Automaton must be hand-encoded; the learning component of their work simply constrains which FSAs that accept a given, comma containing, text span may co-occur. Third, their corpus is preprocessed by hand to identify specialized phrase types needed by their FSAs; once our system has been trained, it can be applied directly to raw text. Fourth, they exclude from their analysis and evaluation any comma they deem to have been incorrectly used in the source text. We include all commas that are present in the

text in our annotation and evaluation.

There is a large body of NLP literature on punctuation. Most of it, however, is concerned with aiding syntactic analysis of sentences and with developing comma checkers, much based on (Nunberg, 1990).

Pattern-based relation extraction methods (e.g., (Davidov and Rappoport, 2008; Davidov et al., 2007; Banko et al., 2007; Pasca et al., 2006; Sekine, 2006)) could in theory be used to extract relations represented by commas. However, the types of patterns used in web-scale lexical approaches currently constrain discovered patterns to relatively short spans of text, so will most likely fail on structures whose arguments cover large spans (for example, appositional clauses containing relative clauses). Relation extraction approaches such as (Roth and Yih, 2004; Roth and Yih, 2007; Hirano et al., 2007; Culotta and Sorenson, 2004; Zelenko et al., 2003) focus on relations between Named Entities; such approaches miss the more general apposition and list relations we recognize in this work, as the arguments in these relations are not confined to Named Entities.

Paraphrase Acquisition work such as that by (Lin and Pantel, 2001; Pantel and Pennacchiotti, 2006; Szpektor et al., 2004) is not constrained to named entities, and by using dependency trees, avoids the locality problems of lexical methods. However, these approaches have so far achieved limited accuracy, and are therefore hard to use to augment existing NLP systems.

### 4 Corpus Annotation

For our corpus, we selected 1,000 sentences containing at least one comma from the Penn Treebank (Marcus et al., 1993) WSJ section 00, and manually annotated them with comma information<sup>3</sup>. This annotated corpus served as both training and test datasets (using cross-validation).

By studying a number of sentences from WSJ (not among the 1,000 selected), we identified four significant types of relations expressed through commas: SUBSTITUTE, ATTRIBUTE, LOCATION, and LIST. Each of these types can in principle be expressed using more than a single comma. We define the notion

<sup>3</sup>The guidelines and annotations are available at <http://L2R.cs.uiuc.edu/~cogcomp/data.php>.

of a *comma structure* as a set of one or more commas that all relate to the same relation in the sentence.

SUBSTITUTE indicates an IS-A relation. An example is ‘John Smith, a Renaissance artist, was famous’. By removing the relation expressed by the commas, we can derive three sentences: ‘John Smith is a Renaissance artist’, ‘John Smith was famous’, and ‘a Renaissance artist was famous’. Note that in theory, the third relation will not be valid: one example is ‘The brothers, all honest men, testified at the trial’, which does not entail ‘all honest men testified at the trial’. However, we encountered no examples of this kind in the corpus, and leave this refinement to future work.

ATTRIBUTE indicates a relation where one argument describes an attribute of the other. For example, from ‘John, who loved chocolate, ate with gusto’, we can derive ‘John loved chocolate’ and ‘John ate with gusto’.

LOCATION indicates a LOCATED-IN relation. For example, from ‘Chicago, Illinois saw some heavy snow today’ we can derive ‘Chicago is located in Illinois’ and ‘Chicago saw some heavy snow today’.

LIST indicates that some predicate or property is applied to multiple entities. In our annotation, the list does not generate explicit relations; instead, the boundaries of the units comprising the list are marked so that they can be treated as a single unit, and are considered to be related by the single relation ‘GROUP’. For example, the derivation of ‘John, James and Kelly all left last week’ is written as ‘[John, James, and Kelly] [all left last week]’.

Any commas not fitting one of the descriptions above are designated as OTHER. This does not indicate that the comma signifies no relations, only that it does not signify a relation of interest in this work (future work will address relations currently subsumed by this category). Analysis of 120 OTHER commas show that approximately half signify clause boundaries, which may occur when sentence constituents are reordered for emphasis, but may also encode implicit temporal, conditional, and other relation types (for example, ‘Opening the drawer, he found the gun.’). The remainder comprises mainly coordination structures (for example, ‘Although he won, he was sad’) and discourse markers indicating inter-sentence relations (such as ‘However, he soon cheered up.’). While we plan to develop an anno-

Rel. Type	Avg. Agreement	# of Commas	# of Rel.s
SUBSTITUTE	0.808	243	729
ATTRIBUTE	0.687	193	386
LOCATION	0.929	71	140
LIST	0.803	230	230
OTHER	0.949	909	0
<i>Combined</i>	0.869	1646	1485

Table 1: Average inter-annotator agreement for identifying relations.

tation scheme for such relations, this is beyond the scope of the present work.

Four annotators annotated the same 10% of the WSJ sentences in order to evaluate inter-annotator agreement. The remaining sentences were divided among the four annotators. The resulting corpus was checked by two judges and the annotation corrected where appropriate; if the two judges disagreed, a third judge was consulted and consensus reached. Our annotators were asked to identify comma structures, and for each structure to write its relation type, its arguments, and all possible simplified version(s) of the original sentence in which the relation implied by the comma has been removed. Arguments must be contiguous units of the sentence and will be referred to as *chunks* hereafter. Agreement statistics and the number of commas and relations of each type are shown in Table 4. The Accuracy closely approximates Kappa score in this case, since the baseline probability of chance agreement is close to zero.

## 5 A Sentence Transformation Rule Learner (ASTRL)

In this section, we describe a new machine learning system that learns Sentence Transformation Rules (STRs) for comma resolution. We first define the hypothesis space (i.e., STRs) and two operations – substitution and introduction. We then define the feature space, motivating the use of Syntactic Parse annotation to learn STRs. Finally, we describe the ASTRL algorithm.

### 5.1 Sentence Transformation Rules

A **Sentence Transformation Rule (STR)** takes a parse tree as input and generates new sentences. We formalize an STR as the pair  $l \rightarrow r$ , where  $l$  is a tree fragment that can consist of non-terminals, POS tags and lexical items.  $r$  is a set  $\{r_i\}$ , each element of which is a template that consists of the non-

terminals of  $l$  and, possibly, some new tokens. This template is used to generate a new sentence, called a *relation*.

The process of applying an STR  $l \rightarrow r$  to a parse tree  $T$  of a sentence  $s$  begins with finding a match for  $l$  in  $T$ . A match is said to be found if  $l$  is a subtree of  $T$ . If matched, the non-terminals of each  $r_i$  are instantiated with the terminals that they cover in  $T$ . Instantiation is followed by generation of the output relations in one of two ways: *introduction* or *substitution*, which is specified by the corresponding  $r_i$ . If an  $r_i$  is marked as an introductory one, then the relation is the terminal sequence obtained by replacing the non-terminals in  $r_i$  with their instantiations. For substitution, firstly, the non-terminals of the  $r_i$  are replaced by their instantiations. The instantiated  $r_i$  replaces all the terminals in  $s$  that are covered by the  $l$ -match. The notions of introduction and substitution were motivated by ideas introduced in (Bar-Haim et al., 2007).

Figure 1 shows an example of an STR and Figure 2 shows the application of this STR to a sentence. In the first relation,  $NP_1$  and  $NP_2$  are instantiated with the corresponding terminals in the parse tree. In the second and third relations, the terminals of  $NP_1$  and  $NP_2$  replace the terminals covered by  $NP_p$ .

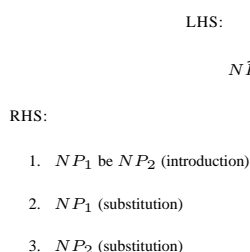
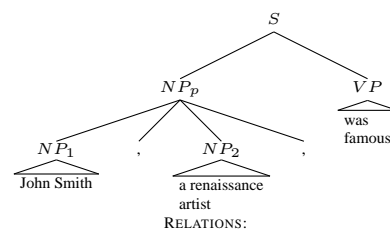


Figure 1: Example of a Sentence Transformation Rule. If the *LHS* matches a part of a given parse tree, then the *RHS* will generate three relations.

## 5.2 The Feature Space

In Section 2, we discussed the example where there could be an ambiguity between a list and an apposition structure in the fragment *two former Soviet republics, Russia and China*. In addition, simple surface examination of the sentence could also identify the noun phrases ‘*Shanghai Co-operation Organization (SCO)*’, ‘*the fledgling association that binds*



- 1 [John Smith]/ $NP_1$  be [a renaissance artist]/ $NP_2$
- 2 [John Smith] / $NP_1$  [was famous]
- 3 [a renaissance artist]/ $NP_2$  [was famous]

Figure 2: Example of application of the STR in Figure 1. In the first relation, an introduction, we use the verb ‘be’, without dealing with its inflections.  $NP_1$  and  $NP_2$  are both substitutions, each replacing  $NP_p$  to generate the last two relations.

*two former Soviet Republics*’, ‘*Russia*’ and ‘*China*’ as the four members of a list. To resolve such ambiguities, we need a nested representation of the sentence. This motivates the use of syntactic parse trees as a logical choice of feature space. (Note, however, that semantic and pragmatic ambiguities might still remain.)

## 5.3 Algorithm Overview

In our corpus annotation, the relations and their argument boundaries (chunks) are explicitly marked. For each training example, our learning algorithm first finds the smallest valid STR – the STR with the smallest *LHS* in terms of depth. Then it refines the *LHS* by specializing it using statistics taken from the entire data set.

## 5.4 Generating the Smallest Valid STR

To transform an example into the smallest valid STR, we utilize the augmented parse tree of the sentence. For each chunk in the sentence, we find the lowest node in the parse tree that covers the chunk and does not cover other chunks (even partially). It may, however, cover words that do not belong to any chunk. We refer to such a node as a *chunk root*. We then find the lowest node that covers all the chunk roots, referring to it as the *pattern root*. The initial *LHS* consists of the subtree of the parse tree rooted at the pattern root and whose leaf nodes are all either chunk roots or nodes that do not belong to any chunk. All the nodes are labeled with the corresponding labels in the aug-

mented parse tree. For example, if we consider the parse tree and relations shown in Figure 2, then doing the above procedure gives us the initial *LHS* as  $S(NP_p(NP_1, NP_2), VP)$ . The three relations gives us the *RHS* with three elements ‘ $NP_1$  be  $NP_2$ ’, ‘ $NP_1 VP$ ’ and ‘ $NP_1 VP$ ’, all three being introduction.

This initial *LHS* need not be the smallest one that explains the example. So, we proceed by finding the lowest node in the initial *LHS* such that the subtree of the *LHS* at that node can form a new STR that covers the example using both introduction *and* substitution. In our example, the initial *LHS* has a subtree,  $NP_p(NP_1, NP_2, )$  that can cover all the relations with the *RHS* consisting of ‘ $NP_1$  be  $NP_2$ ’,  $NP_1$  and  $NP_2$ . The first *RHS* is an introduction, while the second and the third are both substitutions. Since no subtree of this *LHS* can generate all three relations even with substitution, this is the required STR. The final step ensures that we have the smallest valid STR at this stage.

## 5.5 Statistical Refinement

The STR generated using the procedure outlined above explains the relations generated by a single example. In addition to covering the relations generated by the example, we wish to ensure that it does not cover erroneous relations by matching any of the other comma types in the annotated data.

---

### Algorithm 1 ASTRL: A Sentence Transformation Rule Learning.

---

```

1: for all  $t$ : Comma type do
2:   Initialize  $STRList[t] = \emptyset$ 
3:    $\mathbf{p} =$  Set of annotated examples of type  $t$ 
4:    $\mathbf{n} =$  Annotated examples of all other types
5:   for all  $x \in \mathbf{p}$  do
6:      $r =$  Smallest Valid STR that covers  $x$ 
7:     Get fringe of  $r.LHS$  using the parse tree
8:      $S = Score(r, \mathbf{p}, \mathbf{n})$ 
9:      $S_{prev} = -\infty$ 
10:    while  $S \neq S_{prev}$  do
11:      if adding some fringe node to  $r.LHS$  causes a significant change in score then
12:        Set  $r =$  New rule that includes that fringe node
13:         $S_{prev} = S$ 
14:         $S = Score(r, \mathbf{p}, \mathbf{n})$ 
15:        Recompute new fringe nodes
16:      end if
17:    end while
18:    Add  $r$  to  $STRList[t]$ 
19:    Remove all examples from  $\mathbf{p}$  that are covered by  $r$ 
20:  end for
21: end for

```

---

For this purpose, we specialize the *LHS* so that it covers as few examples from the other comma types as possible, while covering as many examples from the current comma type as possible. Given the most general STR, we generate a set of additional, more detailed, candidate rules. Each of these is obtained from the original rule by adding a single node to the tree pattern in the rule’s *LHS*, and updating the rule’s *RHS* accordingly. We then score each of the candidates (including the original rule). If there is a clear winner, we continue with it using the same procedure (i.e., specialize it). If there isn’t a clear winner, we stop and use the current winner. After finishing with a rule (line 18), we remove from the set of positive examples of its comma type all examples that are covered by it (line 19).

To generate the additional candidate rules that we add, we define the *fringe* of a rule as the siblings and children of the nodes in its *LHS* in the original parse tree. Each fringe node defines an additional candidate rule, whose *LHS* is obtained by adding the fringe node to the rule’s *LHS* tree. We refer to the set of these candidate rules, plus the original one, as the rule’s *fringe rules*. We define the score of an STR as

$$Score(Rule, \mathbf{p}, \mathbf{n}) = \frac{R_p}{|\mathbf{p}|} - \frac{R_n}{|\mathbf{n}|}$$

where  $\mathbf{p}$  and  $\mathbf{n}$  are the set of positive and negative examples for this comma type, and  $R_p$  and  $R_n$  are the number of positive and negative examples that are covered by the STR. For each example, all examples annotated with the same comma type are positive while all examples of all other comma types are negative. The score is used to select the winner among the fringe rules. The complete algorithm we have used is listed in Algorithm 1. For convenience, the algorithm’s main loop is given in terms of comma types, although this is not strictly necessary. The stopping criterion in line 11 checks whether any fringe rule has a significantly better score than the rule it was derived from, and exits the specialization loop if there is none.

Since we start with the smallest STR, we only need to add nodes to it to refine it and never have to delete any nodes from the tree. Also note that the algorithm is essentially a greedy algorithm that performs a single pass over the examples; other, more

complex, search strategies could also be used.

## 6 Evaluation

### 6.1 Experimental Setup

To evaluate ASTRL, we used the WSJ derived corpus. We experimented with three scenarios; in two of them we trained using the gold standard trees and then tested on gold standard parse trees (*Gold-Gold*), and text annotated using a state-of-the-art statistical parser (Charniak and Johnson, 2005) (*Gold-Charniak*), respectively. In the third, we trained and tested on the Charniak Parser (*Charniak-Charniak*).

In gold standard parse trees the syntactic categories are annotated with functional tags. Since current statistical parsers do not annotate sentences with such tags, we augment the syntactic trees with the output of a Named Entity tagger. For the Named Entity information, we used a publicly available NE Recognizer capable of recognizing a range of categories including Person, Location and Organization. On the CoNLL-03 shared task, its f-score is about 90%<sup>4</sup>. We evaluate our system from different points of view, as described below. For all the evaluation methods, we performed five-fold cross validation and report the average precision, recall and f-scores.

### 6.2 Relation Extraction Performance

Firstly, we present the evaluation of the performance of ASTRL from the point of view of relation extraction. After learning the STRs for the different comma types using the gold standard parses, we generated relations by applying the STRs on the test set once. Table 2 shows the precision, recall and f-score of the relations, without accounting for the comma type of the STR that was used to generate them. This metric, called the *Relation metric* in further discussion, is the most relevant one from the point of view of the TE task. Since a list does not generate any relations in our annotation scheme, we use the commas to identify the list elements. Treating each list in a sentence as a single relation, we score the list with the fraction of its correctly identified elements.

In addition to the Gold-Gold and Gold-Charniak

<sup>4</sup>A web demo of the NER is at <http://L2R.cs.uiuc.edu/~cogcomp/demos.php>.

settings described above, for this metric, we also present the results of the Charniak-Charniak setting, where both the train and test sets were annotated with the output of the Charniak parser. The improvement in recall in this setting over the Gold-Charniak case indicates that the parser makes systematic errors with respect to the phenomena considered.

Setting	P	R	F
Gold-Gold	86.1	75.4	80.2
Gold-Charniak	77.3	60.1	68.1
Charniak-Charniak	77.2	64.8	70.4

Table 2: ASTRL performance (precision, recall and f-score) for relation extraction. The comma types were used only to learn the rules. During evaluation, only the relations were scored.

### 6.3 Comma Resolution Performance

We present a detailed analysis of the performance of the algorithm for comma resolution. Since this paper is the first one that deals with the task, we could not compare our results to previous work. Also, there is no clear baseline to use. We tried a variant of the most frequent baseline common in other disambiguation tasks, in which we labeled all commas as OTHER (the most frequent type) except when there are list indicators like *and*, *or* and *but* in adjacent chunks (which are obtained using a shallow parser), in which case the commas are labeled LIST. This gives an average precision 0.85 and an average recall of 0.36 for identifying the comma type. However, this baseline does not help in identifying relations.

We use the following approach to evaluate the comma type resolution *and* relation extraction performance – a relation extracted by the system is considered correct only if both the relation and the type of the comma structure that generated it are correctly identified. We call this metric the *Relation-Type metric*. Another way of measuring the performance of comma resolution is to measure the correctness of the relations per comma type. In both cases, lists are scored as in the Relation metric. The performance of our system with respect to these two metrics are presented in Table 3. In this table, we also compare the performance of the STRs learned by ASTRL with the smallest valid STRs without further specialization (i.e., using just the procedure outlined in Section 5.4).



Type	Gold-Gold Setting						Gold-Charniak Setting					
	Relation-Type metric											
	Smallest Valid STRs			ASTRL			Smallest Valid STRs			ASTRL		
	P	R	F	P	R	F	P	R	F	P	R	F
Total	66.2	76.1	70.7	81.8	73.9	<b>77.6</b>	61.0	58.4	59.5	72.2	59.5	<b>65.1</b>
Relations Metric, Per Comma Type												
ATTRIBUTE	40.4	68.2	50.4	70.6	59.4	<b>64.1</b>	35.5	39.7	36.2	56.6	37.7	<b>44.9</b>
SUBSTITUTE	80.0	84.3	81.9	87.9	84.8	<b>86.1</b>	75.8	72.9	74.3	78.0	76.1	<b>76.9</b>
LIST	70.9	58.1	63.5	76.2	57.8	<b>65.5</b>	58.7	53.4	55.6	65.2	53.3	<b>58.5</b>
LOCATION	93.8	86.4	<b>89.1</b>	93.8	86.4	<b>89.1</b>	70.3	37.2	<b>47.2</b>	70.3	37.2	<b>47.2</b>

Table 3: Performance of STRs learned by ASTRL and the smallest valid STRs in identifying comma types and generating relations.

There is an important difference between the Relation metric (Table 2) and the Relation-type metric (top part of Table 3) that depends on the semantic interpretation of the comma types. For example, consider the sentence ‘John Smith, 59, went home.’ If the system labels the commas in this as both ATTRIBUTE and SUBSTITUTE, then, both will generate the relation ‘John Smith is 59.’ According to the Relation metric, there is no difference between them. However, there is a semantic difference between the two sentences – the ATTRIBUTE relation says that being 59 is an attribute of John Smith while the SUBSTITUTE relation says that John Smith is the number 59. This difference is accounted for by the Relation-Type metric.

From this standpoint, we can see that the specialization step performed in the full ASTRL algorithm greatly helps in disambiguating between the ATTRIBUTE and SUBSTITUTE types and consequently, the Relation-Type metric shows an error reduction of 23.5% and 13.8% in the Gold-Gold and Gold-Charniak settings respectively. In the Gold-Gold scenario the performance of ASTRL is much better than in the Gold-Charniak scenario. This reflects the non-perfect performance of the parser in annotating these sentences (parser F-score of 90%).

Another key evaluation question is the performance of the method in identification of the OTHER category. A comma is judged to be as OTHER if no STR in the system applies to it. The performance of ASTRL in this aspect is presented in Table 4. The categorization of this category is important if we wish to further classify the OTHER commas into finer categories.

Setting	P	R	F
Gold-Gold	78.9	92.8	85.2
Gold-Charniak	72.5	92.2	81.2

Table 4: ASTRL performance (precision, recall and f-score) for OTHER identification.

## 7 Conclusions

We defined the task of comma resolution, and developed a novel machine learning algorithm that learns Sentence Transformation Rules to perform this task. We experimented with both gold standard and parser annotated sentences, and established a performance level that seems good for a task of this complexity, and which will provide a useful measure of future systems developed for this task. When given automatically parsed sentences, performance degrades but is still much higher than random, in both scenarios. We designed a comma annotation scheme, where each comma unit is assigned one of four types and an inference rule mapping the patterns of the unit with the entailed relations. We created annotated datasets which will be made available over the web to facilitate further research.

Future work will investigate four main directions: (i) studying the effects of inclusion of our approach on the performance of Textual Entailment systems; (ii) using features other than those derivable from syntactic parse and named entity annotation of the input sentence; (iii) recognizing a wider range of implicit relations, represented by commas and in other ways; (iv) adaptation to other domains.

## Acknowledgement

The UIUC authors were supported by NSF grant ITR IIS-0428472, DARPA funding under the Bootstrap Learning Program and a grant from Boeing.

## References

- M. Banko, M. Cafarella, M. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *Proc. of IJCAI*, pages 2670–2676.
- R. Bar-Haim, I. Dagan, I. Greenal, and E. Shnarch. 2007. Semantic inference at the lexical-syntactic level. In *Proc. of AAAI*, pages 871–876.
- M. Bayraktar, B. Say, and V. Akman. 1998. An analysis of english punctuation: The special case of comma. *International Journal of Corpus Linguistics*, 3(1):33–57.
- E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of the Annual Meeting of the ACL*, pages 173–180.
- A. Culotta and J. Sorenson. 2004. Dependency tree kernels for relation extraction. In *Proc. of the Annual Meeting of the ACL*, pages 423–429.
- I. Dagan, O. Glickman, and B. Magnini, editors. 2006. *The PASCAL Recognising Textual Entailment Challenge.*, volume 3944. Springer-Verlag, Berlin.
- D. Davidov and A. Rappoport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated sat analogy questions. In *Proc. of the Annual Meeting of the ACL*.
- D. Davidov, A. Rappoport, and M. Koppel. 2007. Fully unsupervised discovery of concept-specific relationships by web mining. In *Proc. of the Annual Meeting of the ACL*, pages 232–239.
- R. de Salvo Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. 2005. An inference model for semantic entailment in natural language. In *Proc. of AAAI*, pages 1678–1679.
- T. Hirano, Y. Matsuo, and G. Kikui. 2007. Detecting semantic relations between named entities in text using contextual features. In *Proc. of the Annual Meeting of the ACL*, pages 157–160.
- D. Lin and P. Pantel. 2001. DIRT: discovery of inference rules from text. In *Proc. of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001*, pages 323–328.
- M. P. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- G. Nunberg. 1990. *CSLI Lecture Notes 18: The Linguistics of Punctuation*. CSLI Publications, Stanford, CA.
- P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proc. of the Annual Meeting of the ACL*, pages 113–120.
- M. Pasca, D. Lin, J. Bigam, A. Lifchits, and A. Jain. 2006. Names and similarities on the web: Fact extraction in the fast lane. In *Proc. of the Annual Meeting of the ACL*, pages 809–816.
- D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics.
- D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.
- B. Say and V. Akman. 1997. Current approaches to punctuation in computational linguistics. *Computers and the Humanities*, 30(6):457–469.
- S. Sekine. 2006. On-demand information extraction. In *Proc. of the Annual Meeting of the ACL*, pages 731–738.
- I. Szpektor, H. Tanev, I. Dagan, and B. Coppola. 2004. Scaling web-based of entailment relations. In *Proc. of EMNLP*, pages 49–56.
- S. van Delden and F. Gomez. 2002. Combining finite state automata and a greedy learning algorithm to determine the syntactic roles of commas. In *Proc. of ICTAI*, pages 293–300.
- D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

# Finding Contradictions in Text

Marie-Catherine de Marneffe, Anna N. Rafferty and Christopher D. Manning

Linguistics Department

Stanford University

Stanford, CA 94305

mcdm@stanford.edu

Computer Science Department

Stanford University

Stanford, CA 94305

{rafferty,manning}@stanford.edu

## Abstract

Detecting conflicting statements is a foundational text understanding task with applications in information analysis. We propose an appropriate definition of contradiction for NLP tasks and develop available corpora, from which we construct a typology of contradictions. We demonstrate that a system for contradiction needs to make more fine-grained distinctions than the common systems for entailment. In particular, we argue for the centrality of event coreference and therefore incorporate such a component based on topicality. We present the first detailed breakdown of performance on this task. Detecting some types of contradiction requires deeper inferential paths than our system is capable of, but we achieve good performance on types arising from negation and antonymy.

## 1 Introduction

In this paper, we seek to understand the ways contradictions occur across texts and describe a system for automatically detecting such constructions. As a foundational task in text understanding (Condoravdi et al., 2003), contradiction detection has many possible applications. Consider applying a contradiction detection system to political candidate debates: by drawing attention to topics in which candidates have conflicting positions, the system could enable voters to make more informed choices between candidates and sift through the amount of available information. Contradiction detection could also be applied to intelligence reports, demonstrating which information may need further verification. In bioinfor-

matics where protein-protein interaction is widely studied, automatically finding conflicting facts about such interactions would be beneficial.

Here, we shed light on the complex picture of contradiction in text. We provide a definition of contradiction suitable for NLP tasks, as well as a collection of contradiction corpora. Analyzing these data, we find contradiction is a rare phenomenon that may be created in different ways; we propose a typology of contradiction classes and tabulate their frequencies. Contradictions arise from relatively obvious features such as antonymy, negation, or numeric mismatches. They also arise from complex differences in the structure of assertions, discrepancies based on world-knowledge, and lexical contrasts.

- (1) Police specializing in explosives defused the rockets. Some 100 people were working inside the plant.
- (2) 100 people were injured.

This pair is contradictory: defused rockets cannot go off, and thus cannot injure anyone. Detecting contradictions appears to be a harder task than detecting entailments. Here, it is relatively easy to identify the lack of entailment: the first sentence involves no injuries, so the second is unlikely to be entailed. Most entailment systems function as weak proof theory (Hickl et al., 2006; MacCartney et al., 2006; Zanzotto et al., 2007), but contradictions require deeper inferences and model building. While mismatching information between sentences is often a good cue of non-entailment (Vanderwende et al., 2006), it is not sufficient for contradiction detection which requires more precise comprehension of the consequences of sentences. Assessing event coreference is also essential: for texts to contradict, they must

refer to the same event. The importance of event coreference was recognized in the MUC information extraction tasks in which it was key to identify scenarios related to the same event (Humphreys et al., 1997). Recent work in text understanding has not focused on this issue, but it must be tackled in a successful contradiction system. Our system includes event coreference, and we present the first detailed examination of contradiction detection performance, on the basis of our typology.

## 2 Related work

Little work has been done on contradiction detection. The PASCAL Recognizing Textual Entailment (RTE) Challenges (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007) focused on textual inference in any domain. Condoravdi et al. (2003) first recognized the importance of handling entailment and contradiction for text understanding, but they rely on a strict logical definition of these phenomena and do not report empirical results. To our knowledge, Harabagiu et al. (2006) provide the first empirical results for contradiction detection, but they focus on specific kinds of contradiction: those featuring negation and those formed by paraphrases. They constructed two corpora for evaluating their system. One was created by overtly negating each entailment in the RTE2 data, producing a balanced dataset (LCC\_negation). To avoid overtraining, negative markers were also added to each non-entailment, ensuring that they did not create contradictions. The other was produced by paraphrasing the hypothesis sentences from LCC\_negation, removing the negation (LCC\_paraphrase): *A hunger strike was not attempted* → *A hunger strike was called off*. They achieved very good performance: accuracies of 75.63% on LCC\_negation and 62.55% on LCC\_paraphrase. Yet, contradictions are not limited to these constructions; to be practically useful, any system must provide broader coverage.

## 3 Contradictions

### 3.1 What is a contradiction?

One standard is to adopt a strict logical definition of contradiction: sentences A and B are contradictory if there is no possible world in which A and B are both true. However, for contradiction detection to be

useful, a looser definition that more closely matches human intuitions is necessary; contradiction occurs when two sentences are extremely unlikely to be true simultaneously. Pairs such as *Sally sold a boat to John* and *John sold a boat to Sally* are tagged as contradictory even though it could be that each sold a boat to the other. This definition captures intuitions of incompatibility, and perfectly fits applications that seek to highlight discrepancies in descriptions of the same event. Examples of contradiction are given in table 1. For texts to be contradictory, they must involve the same event. Two phenomena must be considered in this determination: implied coreference and embedded texts. Given limited context, whether two entities are coreferent may be probable rather than certain. To match human intuitions, compatible noun phrases between sentences are assumed to be coreferent in the absence of clear countervailing evidence. In the following example, it is not necessary that the *woman* in the first and second sentences is the same, but one would likely assume it is if the two sentences appeared together:

- (1) Passions surrounding Germany’s final match turned violent when a woman stabbed her partner because she didn’t want to watch the game.
- (2) A woman passionately wanted to watch the game.

We also mark as contradictions pairs reporting contradictory statements. The following sentences refer to the same event (*de Menezes in a subway station*), and display incompatible views of this event:

- (1) Eyewitnesses said de Menezes had jumped over the turnstile at Stockwell subway station.
- (2) The documents leaked to ITV News suggest that Menezes walked casually into the subway station.

This example contains an “embedded contradiction.” Contrary to Zaenen et al. (2005), we argue that recognizing embedded contradictions is important for the application of a contradiction detection system: if *John thinks that he is incompetent*, and *his boss believes that John is not being given a chance*, one would like to detect that the targeted information in the two sentences is contradictory, even though the two sentences can be true simultaneously.

### 3.2 Typology of contradictions

Contradictions may arise from a number of different constructions, some overt and others that are com-

ID	Type	Text	Hypothesis
1	Antonym	Capital punishment is a catalyst for more crime.	Capital punishment is a deterrent to crime.
2	Negation	A closely divided Supreme Court said that juries and not judges must impose a death sentence.	The Supreme Court decided that only judges can impose the death sentence.
3	Numeric	The tragedy of the explosion in Qana that killed more than 50 civilians has presented Israel with a dilemma.	An investigation into the strike in Qana found 28 confirmed dead thus far.
4	Factive	Prime Minister John Howard says he will not be swayed by a warning that Australia faces more terrorism attacks unless it withdraws its troops from Iraq.	Australia withdraws from Iraq.
5	Factive	The bombers had not managed to enter the embassy.	The bombers entered the embassy.
6	Structure	Jacques Santer succeeded Jacques Delors as president of the European Commission in 1995.	Delors succeeded Santer in the presidency of the European Commission.
7	Structure	The Channel Tunnel stretches from England to France. It is the second-longest rail tunnel in the world, the longest being a tunnel in Japan.	The Channel Tunnel connects France and Japan.
8	Lexical	The Canadian parliament's Ethics Commission said former immigration minister, Judy Sgro, did nothing wrong and her staff had put her into a conflict of interest.	The Canadian parliament's Ethics Commission accuses Judy Sgro.
9	Lexical	In the election, Bush called for U.S. troops to be withdrawn from the peacekeeping mission in the Balkans.	He cites such missions as an example of how America must "stay the course."
10	WK	Microsoft Israel, one of the first Microsoft branches outside the USA, was founded in 1989.	Microsoft was established in 1989.

Table 1: Examples of contradiction types.

plex even for humans to detect. Analyzing contradiction corpora (see section 3.3), we find two primary categories of contradiction: (1) those occurring via antonymy, negation, and date/number mismatch, which are relatively simple to detect, and (2) contradictions arising from the use of factive or modal words, structural and subtle lexical contrasts, as well as world knowledge (WK).

We consider contradictions in category (1) ‘easy’ because they can often be automatically detected without full sentence comprehension. For example, if words in the two passages are antonyms and the sentences are reasonably similar, especially in polarity, a contradiction occurs. Additionally, little external information is needed to gain broad coverage of antonymy, negation, and numeric mismatch contradictions; each involves only a closed set of words or data that can be obtained using existing resources and techniques (e.g., WordNet (Fellbaum, 1998), VerbOcean (Chklovski and Pantel, 2004)).

However, contradictions in category (2) are more difficult to detect automatically because they require precise models of sentence meaning. For instance,

to find the contradiction in example 8 (table 1), it is necessary to learn that *X said Y did nothing wrong* and *X accuses Y* are incompatible. Presently, there exist methods for learning oppositional terms (Marcu and Echihabi, 2002) and paraphrase learning has been thoroughly studied, but successfully extending these techniques to learn incompatible phrases poses difficulties because of the data distribution. Example 9 provides an even more difficult instance of contradiction created by a lexical discrepancy. Structural issues also create contradictions (examples 6 and 7). Lexical complexities and variations in the function of arguments across verbs can make recognizing these contradictions complicated. Even when similar verbs are used and argument differences exist, structural differences may indicate non-entailment or contradiction, and distinguishing the two automatically is problematic. Consider contradiction 7 in table 1 and the following non-contradiction:

- (1) The CFAP purchases food stamps from the government and distributes them to eligible recipients.
- (2) A government purchases food.

Data	# contradictions	# total pairs
RTE1_dev1	48	287
RTE1_dev2	55	280
RTE1_test	149	800
RTE2_dev	111	800
RTE3_dev	80	800
RTE3_test	72	800

Table 2: Number of contradictions in the RTE datasets.

In both cases, the first sentence discusses one entity (*CFAP*, *The Channel Tunnel*) with a relationship (*purchase*, *stretch*) to other entities. The second sentence posits a similar relationship that includes one of the entities involved in the original relationship as well as an entity that was not involved. However, different outcomes result because a tunnel connects only two unique locations whereas more than one entity may purchase food. These frequent interactions between world-knowledge and structure make it hard to ensure that any particular instance of structural mismatch is a contradiction.

### 3.3 Contradiction corpora

Following the guidelines above, we annotated the RTE datasets for contradiction. These datasets contain pairs consisting of a short text and a one-sentence hypothesis. Table 2 gives the number of contradictions in each dataset. The RTE datasets are balanced between entailments and non-entailments, and even in these datasets targeting inference, there are few contradictions. Using our guidelines, RTE3\_test was annotated by NIST as part of the RTE3 Pilot task in which systems made a 3-way decision as to whether pairs of sentences were entailed, contradictory, or neither (Voorhees, 2008).<sup>1</sup>

Our annotations and those of NIST were performed on the original RTE datasets, contrary to Harabagiu et al. (2006). Because their corpora are constructed using negation and paraphrase, they are unlikely to cover all types of contradictions in section 3.2. We might hypothesize that rewriting explicit negations commonly occurs via the substitution of antonyms. Imagine, e.g.:

H: Bill has finished his math.

<sup>1</sup>Information about this task as well as data can be found at <http://nlp.stanford.edu/RTE3-pilot/>.

Type	RTE sets	'Real' corpus
1	Antonym	15.0
	Negation	8.8
	Numeric	8.8
2	Factive/Modal	5.0
	Structure	16.3
	Lexical	18.8
	WK	27.5

Table 3: Percentages of contradiction types in the RTE3\_dev dataset and the real contradiction corpus.

Neg-H: Bill hasn't finished his math.

Para-Neg-H: Bill is still working on his math.

The rewriting in both the negated and the paraphrased corpora is likely to leave one in the space of 'easy' contradictions and addresses fewer than 30% of contradictions (table 3). We contacted the LCC authors to obtain their datasets, but they were unable to make them available to us. Thus, we simulated the LCC\_negation corpus, adding negative markers to the RTE2 test data (Neg\_test), and to a development set (Neg\_dev) constructed by randomly sampling 50 pairs of entailments and 50 pairs of non-entailments from the RTE2 development set.

Since the RTE datasets were constructed for textual inference, these corpora do not reflect 'real-life' contradictions. We therefore collected contradictions 'in the wild.' The resulting corpus contains 131 contradictory pairs: 19 from newswire, mainly looking at related articles in Google News, 51 from Wikipedia, 10 from the Lexis Nexis database, and 51 from the data prepared by LDC for the distillation task of the DARPA GALE program. Despite the randomness of the collection, we argue that this corpus best reflects naturally occurring contradictions.<sup>2</sup>

Table 3 gives the distribution of contradiction types for RTE3\_dev and the real contradiction corpus. Globally, we see that contradictions in category (2) occur frequently and dominate the RTE development set. In the real contradiction corpus, there is a much higher rate of the negation, numeric and lexical contradictions. This supports the intuition that in the real world, contradictions primarily occur for two reasons: information is updated as knowledge

<sup>2</sup>Our corpora—the simulation of the LLC\_negation corpus, the RTE datasets and the real contradictions—are available at <http://nlp.stanford.edu/projects/contradiction>.

of an event is acquired over time (e.g., a rising death toll) or various parties have divergent views of an event (e.g., example 9 in table 1).

## 4 System overview

Our system is based on the stage architecture of the Stanford RTE system (MacCartney et al., 2006), but adds a stage for event coreference decision.

### 4.1 Linguistic analysis

The first stage computes linguistic representations containing information about the semantic content of the passages. The text and hypothesis are converted to typed dependency graphs produced by the Stanford parser (Klein and Manning, 2003; de Marneffe et al., 2006). To improve the dependency graph as a pseudo-semantic representation, collocations in WordNet and named entities are collapsed, causing entities and multiword relations to become single nodes.

### 4.2 Alignment between graphs

The second stage provides an alignment between text and hypothesis graphs, consisting of a mapping from each node in the hypothesis to a unique node in the text or to null. The scoring measure uses node similarity (irrespective of polarity) and structural information based on the dependency graphs. Similarity measures and structural information are combined via weights learned using the passive-aggressive online learning algorithm MIRA (Crammer and Singer, 2001). Alignment weights were learned using manually annotated RTE development sets (see Chambers et al., 2007).

### 4.3 Filtering non-coreferent events

Contradiction features are extracted based on mismatches between the text and hypothesis. Therefore, we must first remove pairs of sentences which do not describe the same event, and thus cannot be contradictory to one another. In the following example, it is necessary to recognize that *Pluto's moon* is not the same as *the moon Titan*; otherwise conflicting diameters result in labeling the pair a contradiction.

T: Pluto's moon, which is only about 25 miles in diameter, was photographed 13 years ago.

H: The moon Titan has a diameter of 5100 kms.

This issue does not arise for textual entailment: elements in the hypothesis not supported by the text lead to non-entailment, regardless of whether the same event is described. For contradiction, however, it is critical to filter unrelated sentences to avoid finding false evidence of contradiction when there is contrasting information about different events.

Given the structure of RTE data, in which the hypotheses are shorter and simpler than the texts, one straightforward strategy for detecting coreferent events is to check whether the root of the hypothesis graph is aligned in the text graph. However, some RTE hypotheses are testing systems' abilities to detect relations between entities (e.g., *John of IBM* . . .  $\rightarrow$  *John works for IBM*). Thus, we do not filter verb roots that are indicative of such relations. As shown in table 4, this strategy improves results on RTE data. For real world data, however, the assumption of directionality made in this strategy is unfounded, and we cannot assume that one sentence will be short and the other more complex. Assuming two sentences of comparable complexity, we hypothesize that modeling topicality could be used to assess whether the sentences describe the same event.

There is a continuum of topicality from the start to the end of a sentence (Firbas, 1971). We thus originally defined the topicality of an NP by  $n^w$  where  $n$  is the  $n$ th NP in the sentence. Additionally, we accounted for multiple clauses by weighting each clause equally; in example 4 in table 1, *Australia* receives the same weight as *Prime Minister* because each begins a clause. However, this weighting was not supported empirically, and we thus use a simpler, unweighted model. The topicality score of a sentence is calculated as a normalized score across all aligned NPs.<sup>3</sup> The text and hypothesis are topically related if either sentence score is above a tuned threshold. Modeling topicality provides an additional improvement in precision (table 4).

While filtering provides improvements in performance, some examples of non-coreferent events are still not filtered, such as:

T: Also Friday, five Iraqi soldiers were killed and nine

<sup>3</sup>Since dates can often be viewed as scene setting rather than what the sentence is about, we ignore these in the model. However, ignoring or including dates in the model creates no significant differences in performance on RTE data.

Strategy	Precision	Recall
No filter	55.10	32.93
Root	61.36	32.93
Root + topic	61.90	31.71

Table 4: Precision and recall for contradiction detection on RTE3\_dev using different filtering strategies.

wounded in a bombing, targeting their convoy near Beiji, 150 miles north of Baghdad.

H: Three Iraqi soldiers also died Saturday when their convoy was attacked by gunmen near Adhaim.

It seems that the real world frequency of events needs to be taken into account. In this case, attacks in Iraq are unfortunately frequent enough to assert that it is unlikely that the two sentences present mismatching information (i.e., different location) about the same event. But compare the following example:

T: President Kennedy was assassinated in Texas.

H: Kennedy’s murder occurred in Washington.

The two sentences refer to one unique event, and the location mismatch renders them contradictory.

#### 4.4 Extraction of contradiction features

In the final stage, we extract contradiction features on which we apply logistic regression to classify the pair as contradictory or not. The feature weights are hand-set, guided by linguistic intuition.

### 5 Features for contradiction detection

In this section, we define each of the feature sets used to capture salient patterns of contradiction.

**Polarity features.** Polarity difference between the text and hypothesis is often a good indicator of contradiction, provided there is a good alignment (see example 2 in table 1). The polarity features capture the presence (or absence) of linguistic markers of negative polarity contexts. These markers are scoped such that words are considered negated if they have a negation dependency in the graph or are an explicit linguistic marker of negation (e.g., simple negation (*not*), downward-monotone quantifiers (*no*, *few*), or restricting prepositions). If one word is negated and the other is not, we may have a polarity difference. This difference is confirmed by checking

that the words are not antonyms and that they lack unaligned prepositions or other context that suggests they do not refer to the same thing. In some cases, negations are propagated onto the governor, which allows one to see that *no bullet penetrated* and a *bullet did not penetrate* have the same polarity.

**Number, date and time features.** Numeric mismatches can indicate contradiction (example 3 in table 1). The numeric features recognize (mis-)matches between numbers, dates, and times. We normalize date and time expressions, and represent numbers as ranges. This includes expression matching (e.g., *over 100* and *200* is not a mismatch). Aligned numbers are marked as mismatches when they are incompatible and surrounding words match well, indicating the numbers refer to the same entity.

**Antonymy features.** Aligned antonyms are a very good cue for contradiction. Our list of antonyms and contrasting words comes from WordNet, from which we extract words with direct antonymy links and expand the list by adding words from the same synset as the antonyms. We also use oppositional verbs from VerbOcean. We check whether an aligned pair of words appears in the list, as well as checking for common antonym prefixes (e.g., *anti*, *un*). The polarity of the context is used to determine if the antonyms create a contradiction.

**Structural features.** These features aim to determine whether the syntactic structures of the text and hypothesis create contradictory statements. For example, we compare the subjects and objects for each aligned verb. If the subject in the text overlaps with the object in the hypothesis, we find evidence for a contradiction. Consider example 6 in table 1. In the text, the subject of *succeed* is *Jacques Santer* while in the hypothesis, *Santer* is the object of *succeed*, suggesting that the two sentences are incompatible.

**Factivity features.** The context in which a verb phrase is embedded may give rise to contradiction, as in example 5 (table 1). Negation influences some factivity patterns: *Bill forgot to take his wallet* contradicts *Bill took his wallet* while *Bill did not forget to take his wallet* does not contradict *Bill took his wallet*. For each text/hypothesis pair, we check the (grand)parent of the text word aligned to the hypothesis verb, and generate a feature based on its factiv-



ity class. Factivity classes are formed by clustering our expansion of the PARC lists of factive, implicative and non-factive verbs (Nairn et al., 2006) according to how they create contradiction.

**Modality features.** Simple patterns of modal reasoning are captured by mapping the text and hypothesis to one of six modalities (*(not\_)possible*, *(not\_)actual*, *(not\_)necessary*), according to the presence of predefined modality markers such as *can* or *maybe*. A feature is produced if the text/hypothesis modality pair gives rise to a contradiction. For instance, the following pair will be mapped to the contradiction judgment (*possible*, *not\_possible*):

T: The trial court may allow the prevailing party reasonable attorney fees as part of costs.

H: The prevailing party may not recover attorney fees.

**Relational features.** A large proportion of the RTE data is derived from information extraction tasks where the hypothesis captures a relation between elements in the text. Using Semgrep, a pattern matching language for dependency graphs, we find such relations and ensure that the arguments between the text and the hypothesis match. In the following example, we detect that *Fernandez* works for *FEMA*, and that because of the negation, a contradiction arises.

T: Fernandez, of FEMA, was on scene when Martin arrived at a FEMA base camp.

H: Fernandez doesn't work for FEMA.

Relational features provide accurate information but are difficult to extend for broad coverage.

## 6 Results

Our contradiction detection system was developed on all datasets listed in the first part of table 5. As test sets, we used RTE1\_test, the independently annotated RTE3\_test, and Neg\_test. We focused on attaining high precision. In a real world setting, it is likely that the contradiction rate is extremely low; rather than overwhelming true positives with false positives, rendering the system impractical, we mark contradictions conservatively. We found reasonable inter-annotator agreement between NIST and our post-hoc annotation of RTE3\_test ( $\kappa = 0.81$ ), showing that, even with limited context, humans tend to

	Precision	Recall	Accuracy
RTE1_dev1	70.37	40.43	–
RTE1_dev2	72.41	38.18	–
RTE2_dev	64.00	28.83	–
RTE3_dev	61.90	31.71	–
Neg_dev	74.07	78.43	75.49
Neg_test	62.97	62.50	62.74
LCC_negation	–	–	75.63
RTE1_test	42.22	26.21	–
RTE3_test	22.95	19.44	–
Avg. RTE3_test	10.72	11.69	–

Table 5: Precision and recall figures for contradiction detection. Accuracy is given for balanced datasets only. ‘LCC\_negation’ refers to performance of Harabagiu et al. (2006); ‘Avg. RTE3\_test’ refers to mean performance of the 12 submissions to the RTE3 Pilot.

agree on contradictions.<sup>4</sup> The results on the test sets show that performance drops on new data, highlighting the difficulty in generalizing from a small corpus of positive contradiction examples, as well as underlining the complexity of building a broad coverage system. This drop in accuracy on the test sets is greater than that of many RTE systems, suggesting that generalizing for contradiction is more difficult than for entailment. Particularly when addressing contradictions that require lexical and world knowledge, we are only able to add coverage in a piecemeal fashion, resulting in improved performance on the development sets but only small gains for the test sets. Thus, as shown in table 6, we achieve 13.3% recall on lexical contradictions in RTE3\_dev but are unable to identify any such contradictions in RTE3\_test. Additionally, we found that the precision of category (2) features was less than that of category (1) features. Structural features, for example, caused us to tag 36 non-contradictions as contradictions in RTE3\_test, over 75% of the precision errors. Despite these issues, we achieve much higher precision and recall than the average submission to the RTE3 Pilot task on detecting contradictions, as shown in the last two lines of table 5.

<sup>4</sup>This stands in contrast with the low inter-annotator agreement reported by Sanchez-Graillet and Poesio (2007) for contradictions in protein-protein interactions. The only hypothesis we have to explain this contrast is the difficulty of scientific material.

	Type	RTE3_dev	RTE3_test
1	Antonym	25.0 (3/12)	42.9 (3/7)
	Negation	71.4 (5/7)	60.0 (3/5)
	Numeric	71.4 (5/7)	28.6 (2/7)
2	Factive/Modal	25.0 (1/4)	10.0 (1/10)
	Structure	46.2 (6/13)	21.1 (4/19)
	Lexical	13.3 (2/15)	0.0 (0/12)
	WK	18.2 (4/22)	8.3 (1/12)

Table 6: Recall by contradiction type.

## 7 Error analysis and discussion

One significant issue in contradiction detection is lack of feature generalization. This problem is especially apparent for items in category (2) requiring lexical and world knowledge, which proved to be the most difficult contradictions to detect on a broad scale. While we are able to find certain specific relationships in the development sets, these features attained only limited coverage. Many contradictions in this category require multiple inferences and remain beyond our capabilities:

T: The Auburn High School Athletic Hall of Fame recently introduced its Class of 2005 which includes 10 members.

H: The Auburn High School Athletic Hall of Fame has ten members.

Of the types of contradictions in category (2), we are best at addressing those formed via structural differences and factive/modal constructions as shown in table 6. For instance, we detect examples 5 and 6 in table 1. However, creating features with sufficient precision is an issue for these types of contradictions. Intuitively, two sentences that have aligned verbs with the same subject and different objects (or vice versa) are contradictory. This indeed indicates a contradiction 55% of the time on our development sets, but this is not high enough precision given the rarity of contradictions.

Another type of contradiction where precision falters is numeric mismatch. We obtain high recall for this type (table 6), as it is relatively simple to determine if two numbers are compatible, but high precision is difficult to achieve due to differences in what numbers may mean. Consider:

T: Nike Inc. said that its profit grew 32 percent, as the company posted broad gains in sales and orders.

H: Nike said orders for footwear totaled \$4.9 billion, including a 12 percent increase in U.S. orders.

Our system detects a mismatch between *32 percent* and *12 percent*, ignoring the fact that one refers to *profit* and the other to *orders*. Accounting for context requires extensive text comprehension; it is not enough to simply look at whether the two numbers are headed by similar words (*grew* and *increase*). This emphasizes the fact that mismatching information is not sufficient to indicate contradiction.

As demonstrated by our 63% accuracy on Neg\_test, we are reasonably good at detecting negation and correctly ascertaining whether it is a symptom of contradiction. Similarly, we handle single word antonymy with high precision (78.9%). Nevertheless, Harabagiu et al.’s performance demonstrates that further improvement on these types is possible; indeed, they use more sophisticated techniques to extract oppositional terms and detect polarity differences. Thus, detecting category (1) contradictions is feasible with current systems.

While these contradictions are only a third of those in the RTE datasets, detecting such contradictions accurately would solve half of the problems found in the real corpus. This suggests that we may be able to gain sufficient traction on contradiction detection for real world applications. Even so, category (2) contradictions must be targeted to detect many of the most interesting examples and to solve the entire problem of contradiction detection. Some types of these contradictions, such as lexical and world knowledge, are currently beyond our grasp, but we have demonstrated that progress may be made on the structure and factive/modal types.

Despite being rare, contradiction is foundational in text comprehension. Our detailed investigation demonstrates which aspects of it can be resolved and where further research must be directed.

## Acknowledgments

This paper is based on work funded in part by the Defense Advanced Research Projects Agency through IBM and by the Disruptive Technology Office (DTO) Phase III Program for Advanced Question Answering for Intelligence (AQUAINT) through Broad Agency Announcement (BAA) N61339-06-R-0034.

## References

- Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, Venice, Italy.
- Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.
- Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of EMNLP-04*.
- Cleo Condoravdi, Dick Crouch, Valeria de Pavia, Reinhard Stolle, and Daniel G. Bobrow. 2003. Entailment, intensionality and text understanding. *Workshop on Text Meaning (2003 May 31)*.
- Koby Crammer and Yoram Singer. 2001. Ultraconservative online algorithms for multiclass problems. In *Proceedings of COLT-2001*.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In Quinonero-Candela et al., editor, *MLCW 2005, LNAI Volume 3944*, pages 177–190. Springer-Verlag.
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06)*.
- Christiane Fellbaum. 1998. *WordNet: an electronic lexical database*. MIT Press.
- Jan Firbas. 1971. On the concept of communicative dynamism in the theory of functional sentence perspective. *Brno Studies in English*, 7:23–47.
- Danilo Giampiccolo, Ido Dagan, Bernardo Magnini, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.
- Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast, and contradiction in text processing. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*.
- Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing textual entailment with LCC’s GROUNDHOG system. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.
- Kevin Humphreys, Robert Gaizauskas, and Saliha Azam. 1997. Event coreference for information extraction. In *Proceedings of the Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts, 35th ACL meeting*.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics*.
- Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the North American Association of Computational Linguistics (NAACL-06)*.
- Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.
- Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. In *Proceedings of ICoS-5*.
- Olivia Sanchez-Graillet and Massimo Poesio. 2007. Discovering contradiction protein-protein interactions in text. In *Proceedings of BioNLP 2007: Biological, translational, and clinical language processing*.
- Lucy Vanderwende, Arul Menezes, and Rion Snow. 2006. Microsoft research at rte-2: Syntactic contributions in the entailment task: an implementation. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.
- Ellen Voorhees. 2008. Contradictions and justifications: Extensions to the textual entailment task. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.
- Annie Zaenen, Lauri Karttunen, and Richard S. Crouch. 2005. Local textual inference: can it be defined or circumscribed? In *ACL 2005 Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.
- Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2007. Shallow semantics in fast textual entailment rule learners. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

# Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs

**Zornitsa Kozareva**  
DLSI, University of Alicante  
Campus de San Vicente  
Alicante, Spain 03080  
zkozareva@dlsi.ua.es

**Ellen Riloff**  
School of Computing  
University of Utah  
Salt Lake City, UT 84112  
riloff@cs.utah.edu

**Eduard Hovy**  
USC Information Sciences Institute  
4676 Admiralty Way  
Marina del Rey, CA 90292-6695  
hovy@isi.edu

## Abstract

We present a novel approach to weakly supervised semantic class learning from the web, using a single powerful hyponym pattern combined with graph structures, which capture two properties associated with pattern-based extractions: *popularity* and *productivity*. Intuitively, a candidate is *popular* if it was discovered many times by other instances in the hyponym pattern. A candidate is *productive* if it frequently leads to the discovery of other instances. Together, these two measures capture not only frequency of occurrence, but also cross-checking that the candidate occurs both near the class name and near other class members. We developed two algorithms that begin with just a class name and one seed instance and then automatically generate a ranked list of new class instances. We conducted experiments on four semantic classes and consistently achieved high accuracies.

## 1 Introduction

Knowing the semantic classes of words (e.g., “trout” is a kind of FISH) can be extremely valuable for many natural language processing tasks. Although some semantic dictionaries do exist (e.g., WordNet (Miller, 1990)), they are rarely complete, especially for large open classes (e.g., classes of people and objects) and rapidly changing categories (e.g., computer technology). (Roark and Charniak, 1998) reported that 3 of every 5 terms generated by their semantic lexicon learner were not present in WordNet. Automatic semantic lexicon acquisition could

be used to enhance existing resources such as WordNet, or to produce semantic lexicons for specialized categories or domains.

A variety of methods have been developed for automatic semantic class identification, under the rubrics of lexical acquisition, hyponym acquisition, semantic lexicon induction, semantic class learning, and web-based information extraction. Many of these approaches employ surface-level patterns to identify words and their associated semantic classes. However, such patterns tend to overgenerate (i.e., deliver incorrect results) and hence require additional filtering mechanisms.

To overcome this problem, we employed one single powerful *doubly-anchored* hyponym pattern to query the web and extract semantic class instances: CLASS\_NAME *such as* CLASS\_MEMBER *and* \*.

We hypothesized that a doubly-anchored pattern, which includes both the class name and a class member, would achieve high accuracy because of its specificity. To address concerns about coverage, we embedded the search in a bootstrapping process. This method produced many correct instances, but despite the highly restrictive nature of the pattern, still produced many incorrect instances. This result led us to explore new ways to improve the accuracy of hyponym patterns without requiring additional training resources.

The main contribution of this work is a novel method for combining hyponym patterns with graph structures that capture two properties associated with pattern extraction: *popularity* and *productivity*. Intuitively, a candidate word (or phrase) is *popular* if it was discovered many times by other words (or

phrases) in a hyponym pattern. A candidate word is *productive* if it frequently leads to the discovery of other words. Together, these two measures capture not only frequency of occurrence, but also cross-checking that the word occurs both near the class name and near other class members.

We present two algorithms that use *hyponym pattern linkage graphs (HPLGs)* to represent popularity and productivity information. The first method uses a dynamically constructed HPLG to assess the popularity of each candidate and steer the bootstrapping process. This approach produces an efficient bootstrapping process that performs reasonably well, but it cannot take advantage of productivity information because of the dynamic nature of the process.

The second method is a two-step procedure that begins with an exhaustive pattern search that acquires popularity and productivity information about candidate instances. The candidates are then ranked based on properties of the HPLG. We conducted experiments with four semantic classes, achieving high accuracies and outperforming the results reported by others who have worked on the same classes.

## 2 Related Work

A substantial amount of research has been done in the area of semantic class learning, under a variety of different names and with a variety of different goals. Given the great deal of similar work in information extraction and ontology learning, we focus here only on techniques for weakly supervised or unsupervised semantic class (i.e., supertype-based) learning, since that is most related to the work in this paper.

Fully unsupervised semantic clustering (e.g., (Lin, 1998; Lin and Pantel, 2002; Davidov and Rappoport, 2006)) has the disadvantage that it may or may not produce the types and granularities of semantic classes desired by a user. Another related line of work is automated ontology construction, which aims to create lexical hierarchies based on semantic classes (e.g., (Caraballo, 1999; Cimiano and Volker, 2005; Mann, 2002)), and learning semantic relations such as meronymy (Berland and Charniak, 1999; Girju et al., 2003).

Our research focuses on semantic lexicon induction, which aims to generate lists of words that be-

long to a given semantic class (e.g., lists of FISH or VEHICLE words). Weakly supervised learning methods for semantic lexicon generation have utilized co-occurrence statistics (Riloff and Shepherd, 1997; Roark and Charniak, 1998), syntactic information (Tanev and Magnini, 2006; Pantel and Ravichandran, 2004; Phillips and Riloff, 2002), lexico-syntactic contextual patterns (e.g., “*resides in <location>*” or “*moved to <location>*”) (Riloff and Jones, 1999; Thelen and Riloff, 2002), and local and global contexts (Fleischman and Hovy, 2002). These methods have been evaluated only on fixed corpora<sup>1</sup>, although (Pantel et al., 2004) demonstrated how to scale up their algorithms for the web.

Several techniques for semantic class induction have also been developed specifically for learning from the web. (Paşca, 2004) uses Hearst’s patterns (Hearst, 1992) to learn semantic class instances and class groups by acquiring contexts around the pattern. Paşca also developed a second technique (Paşca, 2007b) that creates context vectors for a group of seed instances by searching web query logs, and uses them to learn similar instances.

The work most closely related to ours is Hearst’s early work on hyponym learning (Hearst, 1992) and more recent work that has followed up on her idea. Hearst’s system exploited patterns that explicitly identify a hyponym relation between a semantic class and a word (e.g., “*such authors as Shakespeare*”). We will refer to these as *hyponym patterns*. Paşca’s previously mentioned system (Paşca, 2004) applies hyponym patterns to the web and acquires contexts around them. The KnowItAll system (Etzioni et al., 2005) also uses hyponym patterns to extract class instances from the web and then evaluates them further by computing mutual information scores based on web queries.

The work by (Widdows and Dorow, 2002) on lexical acquisition is similar to ours because they also use graph structures to learn semantic classes. However, their graph is based entirely on syntactic relations between words, while our graph captures the ability of instances to find each other in a hyponym pattern based on web querying, without any part-of-speech tagging or parsing.

<sup>1</sup>Meta-bootstrapping (Riloff and Jones, 1999) was evaluated on web pages, but used a precompiled corpus of downloaded web pages.

### 3 Semantic Class Learning with Hyponym Pattern Linkage Graphs

#### 3.1 A Doubly-Anchored Hyponym Pattern

Our work was motivated by early research on hyponym learning (Hearst, 1992), which applied patterns to a corpus to associate words with semantic classes. Hearst’s system exploited patterns that explicitly link a class name with a class member, such as “*X and other Ys*” and “*Ys such as X*”. Relying on surface-level patterns, however, is risky because incorrect items are frequently extracted due to polysemy, idiomatic expressions, parsing errors, etc.

Our work began with the simple idea of using an *extremely* specific pattern to extract semantic class members with high accuracy. Our expectation was that a very specific pattern would virtually eliminate the most common types of false hits that are caused by phenomena such as polysemy and idiomatic expressions. A concern, however, was that an extremely specific pattern would suffer from sparse data and not extract many new instances. By using the web as a corpus, we hoped that the pattern could extract at least a few instances for virtually any class, and then we could gain additional traction by bootstrapping these instances.

All of the work presented in this paper uses just one *doubly-anchored* pattern to identify candidate instances for a semantic class:

*<class\_name> such as <class\_member> and \**

This pattern has two variables: the name of the semantic class to be learned (*class\_name*) and a member of the semantic class (*class\_member*). The asterisk (\*) indicates the location of the extracted words. We describe this pattern as being *doubly-anchored* because it is instantiated with both the name of the semantic class as well as a class member.

For example, the pattern “*CARS such as FORD and \**” will extract automobiles, and the pattern “*PRESIDENTS such as FORD and \**” will extract presidents. The doubly-anchored nature of the pattern serves two purposes. First, it increases the likelihood of finding a true list construction for the class. Our system does not use part-of-speech tagging or parsing, so the pattern itself is the only guide for finding an appropriate linguistic context.

Second, the doubly-anchored pattern virtually

```
Members = {Seed};
P0 = “Class such as Seed and *”;
P = {P0};
iter = 0;
While ((iter < Max_Iters) and (P ≠ {}))
  iter++;
  For each Pi ∈ P
    Snippets = web_query(Pi);
    Candidates = extract_words(Snippets, Pi);
    Pnew = {};
    For each Candidatek ∈ Candidates
      If (Candidatek ∉ Members);
        Members = Members ∪ {Candidatek};
        Pk = “Class such as Candidatek and *”;
        Pnew = Pnew ∪ { Pk };
    P = Pnew;
```

Figure 1: Reckless Bootstrapping

eliminates ambiguity because the *class\_name* and *class\_member* mutually disambiguate each other. For example, the word FORD could refer to an automobile or a person, but in the pattern “*CARS such as FORD and \**” it will almost certainly refer to an automobile. Similarly, the class “PRESIDENT” could refer to country presidents or corporate presidents, and “BUSH” could refer to a plant or a person. But in the pattern “*PRESIDENTS such as BUSH*”, both words will surely refer to country presidents.

Another advantage of the doubly-anchored pattern is that an ambiguous or underspecified class name will be constrained by the presence of the class member. For example, to generate a list of company presidents, someone might naively define the class name as PRESIDENTS. A singly-anchored pattern (e.g., “*PRESIDENTS such as \**”) might generate lists of other types of presidents (e.g., country presidents, university presidents, etc.). Because the doubly-anchored pattern also requires a class member (e.g., “*PRESIDENTS such as BILL GATES and \**”), it is likely to generate only the desired types of instances.

#### 3.2 Reckless Bootstrapping

To evaluate the performance of the doubly-anchored pattern, we began by using the pattern to search the web and embedded this process in a simple bootstrapping loop, which is presented in Figure 1. As input, the user must provide the name of the desired

semantic class (*Class*) and a seed example (*Seed*), which are used to instantiate the pattern. On the first iteration, the pattern is given to Google as a web query, and new class members are extracted from the retrieved text snippets. We wanted the system to be as language-independent as possible, so we refrained from using any taggers or parsing tools. As a result, instances are extracted using only word boundaries and orthographic information. For proper name classes, we extract all capitalized words that immediately follow the pattern. For common noun classes, we extract just one word, if it is not capitalized. Examples are shown below, with the extracted items underlined:

*countries such as China and Sri Lanka are ...*  
*fishes such as trout and bass can ...*

One limitation is that our system cannot learn multi-word instances of common noun categories, or proper names that include uncapitalized words (e.g., “United States of America”). These limitations could be easily overcome by incorporating a noun phrase (NP) chunker and extracting NPs.

Each new class member is then used as a seed instance in the bootstrapping loop. We implemented this process as breadth-first search, where each “ply” of the search process is the result of bootstrapping the class members learned during the previous iteration as seed instances for the next one. During each iteration, we issue a new web query and add the newly extracted class members to the queue for the next cycle. We run this bootstrapping process for a fixed number of iterations (search ply), or until no new class members are produced. We will refer to this process as *reckless bootstrapping* because there are no checks of any kind. Every term extracted by the pattern is assumed to be a class member.

### 3.2.1 Results

Table 1 shows the results for 4 iterations of reckless bootstrapping for four semantic categories: *U.S. states*, *countries*, *singers*, and *fish*. The first two categories are relatively small, closed sets (our gold standard contains 50 U.S. states and 194 countries). The *singers* and *fish* categories are much larger, open sets (see Section 4 for details).

Table 1 reveals that the doubly-anchored pattern achieves high accuracy during the first iteration, but

Iter.	<i>countries</i>	<i>states</i>	<i>singers</i>	<i>fish</i>
1	.80	.79	.91	.76
2	.57	.21	.87	.64
3	.21	.18	.86	.54
4	.16	–	.83	.54

Table 1: *Reckless Bootstrapping Accuracies*

quality deteriorates rapidly as bootstrapping progresses. Figure 2 shows the recall and precision curves for countries and states. High precision is achieved only with low levels of recall for countries. Our initial hypothesis was that such a specific pattern would be able to maintain high precision because non-class members would be unlikely to co-occur with the pattern. But we were surprised to find that many incorrect entries were generated for reasons such as broken expressions like “Mercede -dez”, misidentified list constructions (e.g., “*In countries such as China U.S. Policy is failing...*”), and incomplete proper names due to insufficient length of the retrieved text snippet.

Incorporating a noun phrase chunker would eliminate some of these cases, but far from all of them. We concluded that even such a restrictive pattern is not sufficient for semantic class learning on its own.

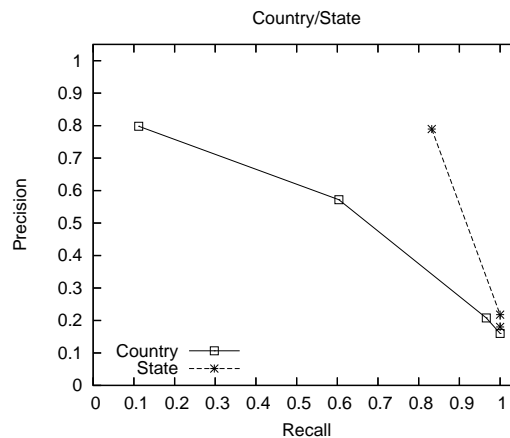


Figure 2: Recall/precision for reckless bootstrapping

In the next section, we present a new approach that creates a Hyponym Pattern Linkage Graph to steer bootstrapping and improve accuracy.

### 3.3 Using Dynamic Graphs to Steer Bootstrapping

Intuitively, we expect true class members to occur frequently in pattern contexts with other class mem-

bers. To operationalize this intuition, we create a *hyponym pattern linkage graph*, which represents the frequencies with which candidate instances generate each other in the pattern contexts.

We define a *hyponym pattern linkage graph (HPLG)* as a  $G = (V, E)$ , where each vertex  $v \in V$  is a candidate instance and each edge  $(u, v) \in E$  means that instance  $v$  was generated by instance  $u$ . The weight  $w$  of an edge is the frequency with which  $u$  generated  $v$ . For example, consider the following sentence, where the pattern is italicized and the extracted instance is underlined:

*Countries such as China and Laos have been...*

In the HPLG, an edge  $e = (China, Laos)$  would be created because the pattern anchored by China extracted Laos as a new candidate instance. If this pattern extracted Laos from 15 different snippets, then the edge’s weight would be 15. The in-degree of a node represents its *popularity*, i.e., the number of instance occurrences that generated it.

The graph is constructed dynamically as bootstrapping progresses. Initially, the seed is the only *trusted class member* and the only vertex in the graph. The bootstrapping process begins by instantiating the doubly-anchored pattern with the seed class member, issuing a web query to generate new candidate instances, and adding these new instances to the graph. A score is then assigned to every node in the graph, using one of several different metrics defined below. The highest-scoring unexplored node is then added to the set of *trusted class members*, and used as the seed for the next bootstrapping iteration.

We experimented with three scoring functions for selecting nodes. The **In-Degree (inD) score** for vertex  $v$  is the sum of the weights of all incoming edges  $(u, v)$ , where  $u$  is a trusted class member. Intuitively, this captures the popularity of  $v$  among instances that have already been identified as good instances. The **Best Edge (BE) score** for vertex  $v$  is the maximum edge weight among the incoming edges  $(u, v)$ , where  $u$  is a trusted class member.

The Key Player Problem (KPP) measure is used in social network analysis (Borgatti and Everett, 2006) to identify nodes whose removal would result in a residual network of minimum cohesion. A node receives a high value if it is highly connected and relatively close to most other nodes in the graph. The

**KPP score** for vertex  $v$  is computed as:

$$KPP(v) = \frac{\sum_{u \in V} \frac{1}{d(u, v)}}{|V|-1}$$

where  $d(u, v)$  is the shortest path between two vertices, where  $u$  is a trusted node. For tie-breaking, the distances are multiplied by the weight of the edge.

Note that all of these measures rely only on incoming edges because a node does not acquire outgoing edges until it has already been selected as a trusted class member and used to acquire new instances. In the next section, we describe a two-step process for creating graphs that can take advantage of both incoming and outgoing edges.

### 3.4 Re-Ranking with Precompiled Graphs

One way to try to confirm (or disconfirm) whether a candidate instance is a true class member is to see whether it can produce new candidate instances. If we instantiate our pattern with the candidate (i.e., “CLASS\_NAME *such as* CANDIDATE *and* \*”) and successfully extract many new instances, then this is evidence that the candidate frequently occurs with the CLASS\_NAME in list constructions. We will refer to the ability of a candidate to generate new instances as its *productivity*.

The previous bootstrapping algorithm uses a dynamically constructed graph that is constantly evolving as new nodes are selected and explored. Each node is scored based only on the set of instances that have been generated and identified as “trusted” at that point in the bootstrapping process. To use productivity information, we must adopt a different procedure because we need to know not only who generated each candidate, but also the complete set of instances that the candidate itself can generate.

We adopted a two-step process that can use both popularity and productivity information in a hyponym pattern linkage graph to assess the quality of candidate instances. First, we perform *reckless bootstrapping* for a *class\_name* and *seed* until no new instances are generated. Second, we assign a score to each node in the graph using a scoring function that takes into account both the in-degree (popularity) and out-degree (productivity) of each node. We experimented with four different scoring functions, some of which were motivated by work on word



sense disambiguation to identify the most “important” node in a graph containing its possible senses (Navigli and Lapata, 2007).

The **Out-degree (outD) score** for vertex  $v$  is the weighted sum of  $v$ ’s outgoing edges, normalized by the number of other nodes in the graph.

$$outD(v) = \frac{\sum_{\forall(v,p) \in E} w(v,p)}{|V|-1}$$

This measure captures only productivity, while the next three measures consider both productivity and popularity. The **Total-degree (totD) score** for vertex  $v$  is the weighted sum of both incoming and outgoing edges, normalized by the number of other nodes in the graph. The **Betweenness (BT) score** (Freeman, 1979) considers a vertex to be important if it occurs on many shortest paths between other vertices.

$$BT(v) = \sum_{s,t \in V: s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where  $\sigma_{st}$  is the number of shortest paths from  $s$  to  $t$ , and  $\sigma_{st}(v)$  is the number of shortest paths from  $s$  to  $t$  that pass through vertex  $v$ . PageRank (Page et al., 1998) establishes the relative importance of a vertex  $v$  through an iterative Markov chain model. The **PageRank (PR) score** of a vertex  $v$  is determined on the basis of the nodes it is connected to.

$$PR(v) = \frac{(1-\alpha)}{|V|} + \alpha \sum_{u,v \in E} \frac{PR(u)}{outdegree(u)}$$

$\alpha$  is a damping factor that we set to 0.85. We discarded all instances that produced zero productivity links, meaning that they did not generate any other candidates when used in web queries.

## 4 Experimental evaluation

### 4.1 Data

We evaluated our algorithms on four semantic categories: *U.S. states*, *countries*, *singers*, and *fish*. The *states* and *countries* categories are relatively small, closed sets: our gold standards consist of 50 U.S. states and 194 countries (based on a list found on Wikipedia). The *singers* and *fish* categories are much larger, open classes. As our gold standard for *fish*, we used a list of common fish names found on Wikipedia.<sup>2</sup> All the singer names generated by our

<sup>2</sup>We also counted as correct plural versions of items found on the list. The total size of our fish list is 1102.

States							
	Popularity			Prd	Pop&Prd		
N	BE	KPP	inD	outD	totD	BT	PR
25	1.0	1.0	1.0	1.0	1.0	.88	.88
50	.96	.98	.98	1.0	1.0	.86	.82
64	.77	.78	.77	.78	.78	.77	.67
Countries							
	Popularity			Prd	Pop&Prd		
N	BE	KPP	inD	outD	totD	BT	PR
50	.98	.97	.98	1.0	1.0	.98	.97
100	.96	.97	.94	1.0	.99	.97	.95
150	.90	.92	.91	1.0	.95	.94	.92
200	.83	.81	.83	.90	.87	.82	.80
300	.60	.59	.61	.61	.62	.56	.60
323	.57	.55	.57	.57	.58	.52	.57
Singers							
	Popularity			Prd	Pop&Prd		
N	BE	KPP	inD	outD	totD	BT	PR
10	.92	.96	.92	1.0	1.0	1.0	1.0
25	.89	.90	.91	1.0	1.0	1.0	.99
50	.92	.85	.92	.97	.98	.95	.97
75	.89	.83	.91	.96	.95	.93	.95
100	.86	.81	.89	.96	.93	.94	.94
150	.86	.79	.88	.95	.92	.93	.87
180	.86	.80	.87	.91	.91	.91	.88
Fish							
	Popularity			Prd	Pop&Prd		
N	BE	KPP	inD	outD	totD	BT	PR
10	.90	.90	.90	1.0	1.0	.90	.70
25	.80	.88	.76	1.0	.96	.96	.72
50	.82	.80	.78	1.0	.94	.88	.66
75	.72	.69	.72	.93	.87	.79	.64
100	.63	.68	.66	.84	.80	.74	.62
116	.60	.65	.66	.80	.78	.71	.59

Table 2: Accuracies for each semantic class

algorithms were manually reviewed for correctness. We evaluated performance in terms of accuracy (the percentage of instances that were correct).<sup>3</sup>

### 4.2 Performance

Table 2 shows the accuracy results of the two algorithms that use hyponym pattern linkage graphs. We display results for the top-ranked N candidates, for all instances that have a productivity value  $>$  zero.<sup>4</sup> The Popularity columns show results for the

<sup>3</sup>We never generated duplicates so the instances are distinct.

<sup>4</sup>Obviously, this cutoff is not available to the popularity-based bootstrapping algorithm, but here we are just comparing the top N results for both algorithms.

bootstrapping algorithm described in Section 3.3, using three different scoring functions. The results for the ranking algorithm described in Section 3.4 are shown in the Productivity (Prd) and Popularity&Productivity (Pop&Prd) columns. For the *states*, *countries*, and *singers* categories, we randomly selected 5 different initial seeds and then averaged the results. For the *fish* category we ran each algorithm using just the seed “salmon”.

The popularity-based metrics produced good accuracies on the *states*, *countries*, and *singers* categories under all 3 scoring functions. For *fish*, KPP performed better than the others.

The *Out-degree (outD)* scoring function, which uses only Productivity information, obtained the best results across all 4 categories. OutD achieved 100% accuracy for the first 50 states and fish, 100% accuracy for the top 150 countries, and 97% accuracy for the top 50 singers. The three scoring metrics that use both popularity and productivity also performed well, but productivity information by itself seems to perform better in some cases.

It can be difficult to compare the results of different semantic class learners because there is no standard set of benchmark categories, so researchers report results for different classes. For the state and country categories, however, we can compare our results with that of other web-based semantic class learners such as Pasca (Paşca, 2007a) and the KnowItAll system (Etzioni et al., 2005). For the U.S. states category, our system achieved 100% recall and 100% precision for the first 50 items generated, and KnowItAll performed similarly achieving 98% recall with 100% precision. Pasca did not evaluate his system on states.

For the *countries* category, our system achieved 100% precision for the first 150 generated instances (77% recall). (Paşca, 2007a) reports results of 100% precision for the first 25 instances generated, and 82% precision for the first 150 instances generated. The KnowItAll system (Etzioni et al., 2005) achieved 97% precision with 58% recall, and 79% precision with 87% recall.<sup>5</sup> To the best of our knowledge, other researchers have not reported results for the singer and fish categories.

<sup>5</sup>(Etzioni et al., 2005) do not report exactly how many countries were in their gold standard.

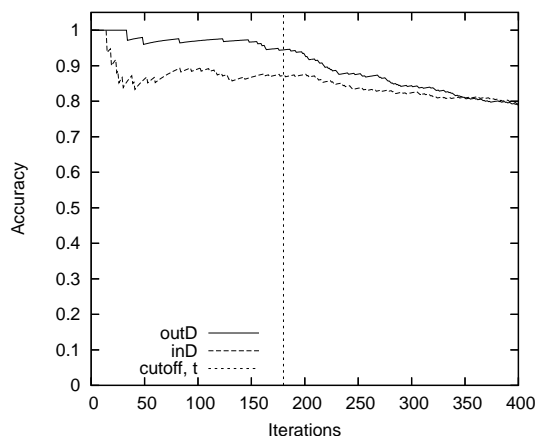


Figure 3: Learning curve for Placido Domingo

Figure 3 shows the learning curve for both algorithms using their best scoring functions on the *singer* category with *Placido Domingo* as the initial seed. In total, 400 candidate words were generated. The *Out-degree* scoring function ranked the candidates well. Figure 3 also includes a vertical line indicating where the candidate list was cut (at 180 instances) based on the zero productivity cutoff.

One observation is that the rankings do a good job of identifying borderline cases, which typically are ranked just below most correct instances but just above the obviously bad entries. For example, for *states*, the 50 U.S. states are ranked first, followed by 14 more entries (in order):

Russia, Ukraine, Uzbekistan, Azerbaijan, Moldova, Tajikistan, Armenia, Chicago, Boston, Atlanta, Detroit, Philadelphia, Tampa, Moldavia

The first 7 entries are all former states of the Soviet Union. In retrospect, we realized that we should have searched for “U.S. states” instead of just “states”. This example illustrates the power of the doubly-anchored hyponym pattern to correctly identify our intended semantic class by disambiguating our class name based on the seed class member.

The algorithms also seem to be robust with respect to initial seed choice. For the *states*, *countries*, and *singers* categories, we ran experiments with 5 different initial seeds, which were randomly selected. The 5 country seeds represented a diverse set of nations, some of which are rarely mentioned in the news: *Brazil*, *France*, *Guinea-Bissau*, *Uganda*,

and *Zimbabwe*. All of these seeds obtained  $\geq 92\%$  recall with  $\geq 90\%$  precision.

### 4.3 Error Analysis

We examined the incorrect instances produced by our algorithms and found that most of them fell into five categories.

Type 1 errors were caused by incorrect proper name extraction. For example, in the sentence “*states such as Georgia and English speaking countries like Canada...*”, “*English*” was extracted as a state. These errors resulted from complex noun phrases and conjunctions, as well as unusual syntactic constructions. An NP chunker might prevent some of these cases, but we suspect that many of them would have been misparsed regardless.

Type 2 errors were caused by instances that formerly belonged to the semantic class (e.g., *Serbia-Montenegro* and *Czechoslovakia* are no longer countries). In this error type, we also include borderline cases that could arguably belong to the semantic class (e.g., *Wales* as a country).

Type 3 errors were spelling variants (e.g., *Kyrgystan* vs. *Kyrgyzstan*) and name variants (e.g., *Beyonce* vs. *Beyonce Knowles*). Officially, every entity has one official spelling and one complete name, but in practice there are often variations that may occur nearly as frequently as the official name. For example, it is most common to refer to the singer *Beyonce* by just her first name.

Type 4 errors were caused by sentences that were just flat out wrong in their factual assertions. For example, some sentences referred to “*North America*” as a country.

Type 5 errors were caused by broken expressions found in the retrieved snippets (e.g. *Michi -gan*). These errors may be fixable by cleaning up the web pages or applying heuristics to prevent or recognize partial words.

It is worth noting that incorrect instances of Types 2 and 3 may not be problematic to encounter in a dictionary or ontology. Name variants and former class members may in fact be useful to have.

## 5 Conclusions

Combining hyponym patterns with pattern linkage graphs is an effective way to produce a highly ac-

curate semantic class learner that requires truly minimal supervision: just the class name and one class member as a seed. Our results consistently produced high accuracy and for the *states* and *countries* categories produced very high recall.

The *singers* and *fish* categories, which are much larger open classes, also achieved high accuracy and generated many instances, but the resulting lists are far from complete. Even on the web, the doubly-anchored hyponym pattern eventually ran out of steam and could not produce more instances. However, all of our experiments were conducted using just a single hyponym pattern. Other researchers have successfully used sets of hyponym patterns (e.g., (Hearst, 1992; Etzioni et al., 2005; Paşca, 2004)), and multiple patterns could be used with our algorithms as well. Incorporating additional hyponym patterns will almost certainly improve coverage, and could potentially improve the quality of the graphs as well.

Our popularity-based algorithm was very effective and is practical to use. Our best-performing algorithm, however, was the 2-step process that begins with an exhaustive search (reckless bootstrapping) and then ranks the candidates using the *Out-degree* scoring function, which represents productivity. The first step is expensive, however, because it exhaustively applies the pattern to the web until no more extractions are found. In our evaluation, we ran this process on a single PC and it usually finished overnight, and we were able to learn a substantial number of new class instances. If more hyponym patterns are used, then this could get considerably more expensive, but the process could be easily parallelized to perform queries across a cluster of machines. With access to a cluster of ordinary PCs, this technique could be used to automatically create extremely large, high-quality semantic lexicons, for virtually any categories, without external training resources.

## Acknowledgments

This research was supported in part by the Department of Homeland Security under ONR Grants N00014-07-1-014 and N0014-07-1-0152, the European Union Sixth Framework project QALLME FP6 IST-033860, and the Spanish Ministry of Science and Technology TEXT-MESS TIN2006-15265-C06-01.

## References

- M. Berland and E. Charniak. 1999. Finding Parts in Very Large Corpora. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*.
- S. Borgatti and M. Everett. 2006. A graph-theoretic perspective on centrality. *Social Networks*, 28(4).
- S. Caraballo. 1999. Automatic Acquisition of a Hypernym-Labeled Noun Hierarchy from Text. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126.
- P. Cimiano and J. Volker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *Proc. of Recent Advances in Natural Language Processing*, pages 166–172.
- D. Davidov and A. Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proc. of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*.
- O. Etzioni, M. Cafarella, D. Downey, 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, June.
- M.B. Fleischman and E.H. Hovy. 2002. Fine grained classification of named entities. In *Proc. of the 19th International Conference on Computational Linguistics*, pages 1–7.
- C. Freeman. 1979. Centrality in social networks: Conceptual clarification. *Social Networks*, 1:215–239.
- R. Girju, A. Badulescu, and D. Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proc. of Conference of HLT / North American Chapter of the Association for Computational Linguistics*.
- M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th conference on Computational linguistics*, pages 539–545.
- D. Lin and P. Pantel. 2002. Concept discovery from text. In *Proc. of the 19th International Conference on Computational linguistics*, pages 1–7.
- D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of the 17th international conference on Computational linguistics*, pages 768–774.
- G. Mann. 2002. Fine-grained proper noun ontologies for question answering. In *Proc. of the 19th International Conference on Computational Linguistics*, pages 1–7.
- G. Miller. 1990. Wordnet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4).
- R. Navigli and M. Lapata. 2007. Graph connectivity measures for unsupervised word sense disambiguation. In *Proc. of the 20th International Joint Conference on Artificial Intelligence*, pages 1683–1688.
- M. Paşca. 2004. Acquisition of categorized named entities for web search. In *Proc. of the Thirteenth ACM International Conference on Information and Knowledge Management*, pages 137–145.
- M. Paşca. 2007a. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *Proc. of the 16th International Conference on World Wide Web*, pages 101–110.
- M. Paşca. 2007b. Weakly-supervised discovery of named entities using web search queries. In *Proc. of the sixteenth ACM conference on Conference on information and knowledge management*, pages 683–690.
- L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.
- P. Pantel and D. Ravichandran. 2004. Automatically labeling semantic classes. In *Proc. of Conference of HLT / North American Chapter of the Association for Computational Linguistics*, pages 321–328.
- P. Pantel, D. Ravichandran, and E. Hovy. 2004. Towards terascale knowledge acquisition. In *Proc. of the 20th international conference on Computational Linguistics*, page 771.
- W. Phillips and E. Riloff. 2002. Exploiting Strong Syntactic Heuristics and Co-Training to Learn Semantic Lexicons. In *Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing*.
- E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proc. of the Sixteenth National Conference on Artificial Intelligence*.
- E. Riloff and J. Shepherd. 1997. A Corpus-Based Approach for Building Semantic Lexicons. In *Proc. of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124.
- B. Roark and E. Charniak. 1998. Noun-phrase Co-occurrence Statistics for Semi-automatic Semantic Lexicon Construction. In *Proc. of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1110–1116.
- H. Tanev and B. Magnini. 2006. Weakly supervised approaches for ontology population. In *Proc. of 11st Conference of the European Chapter of the Association for Computational Linguistics*.
- M. Thelen and E. Riloff. 2002. A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pattern Contexts. In *Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing*.
- D. Widdows and B. Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proc. of the 19th International Conference on Computational Linguistics*, pages 1–7.

# Author Index

- Adams, Jeff, 114  
Adler, Meni, 728, 746  
Agirre, Eneko, 317, 550  
Ai, Hua, 622  
Andreevskaia, Alina, 290  
Arnold, Andrew, 245  
Avramidis, Eleftherios, 763  
Aw, Aiti, 559
- Baldrige, Jason, 326  
Baldwin, Timothy, 317  
Banko, Michele, 28  
Bao, Shenghua, 914  
Bar-Haim, Roy, 683  
Bartlett, Susan, 568  
Barzilay, Regina, 263, 737, 852  
Bender, Emily M., 977  
Berg-Kirkpatrick, Taylor, 771  
Bergler, Sabine, 290  
Bergsma, Shane, 10  
Bhagat, Rahul, 674  
Biadsy, Fadi, 807  
Bisani, Maximilian, 114  
Blunsom, Phil, 200  
Branavan, S.R.K., 263  
Brants, Thorsten, 505, 755  
Brew, Chris, 434
- Cao, Guihong, 148  
Cao, Yunbo, 156, 914  
Carenini, Giuseppe, 353  
Carreras, Xavier, 595  
Carroll, John, 968  
Chambers, Nathanael, 789  
Chan, Yee Seng, 55  
Charniak, Eugene, 834  
Chen, Harr, 263  
Cherry, Colin, 72, 568, 905  
Ciaramita, Massimiliano, 719
- Clark, Stephen, 888  
Cohen, William W., 245  
Cohn, Trevor, 200  
Collins, Michael, 595  
Cong, Gao, 710  
Csomai, Andras, 932  
Curran, James R., 335
- Dagan, Ido, 683  
Daumé III, Hal, 389  
Davidov, Dmitry, 227, 692  
Davis, Randall, 852  
de Marneffe, Marie-Catherine, 1039  
de Rijke, Maarten, 923  
Deng, Yonggang, 81  
Diab, Mona, 798  
Dickinson, Markus, 362  
Ding, Shilin, 710  
Divay, Olivier, 114  
dos Santos, Cícero Nogueira, 647  
Dridan, Rebecca, 613  
Duan, Huizhong, 156, 914  
Duan, Nan, 89  
Duarte, Julio C., 647  
Dyer, Christopher, 1012
- Eisenstein, Jacob, 263, 852  
Elhadad, Michael, 728, 746  
Elsayed, Tamer, 941  
Elsner, Micha, 834  
Espinosa, Dominic, 183  
Etzioni, Oren, 28
- Fang, Hui, 139  
Feng, Yansong, 272  
Filatova, Elena, 807  
Finkel, Jenny Rose, 959  
Fleck, Margaret M., 130  
Fleischman, Michael, 121

Furui, Sadaoki, 443

Gabay, David, 728

Ganchev, Kuzman, 986

Gao, Yuqing, 81

Gildea, Daniel, 97, 209

Goebel, Randy, 10

Goldberg, Andrew B., 656

Goldberg, Yoav, 371, 728, 746

Goldberger, Jacob, 683

Goldwater, Sharon, 380

Gómez-Rodríguez, Carlos, 968

Graça, João V., 986

Grishman, Ralph, 254

Haghighi, Aria, 771

Hahn, Udo, 861

Hao, Yanfen, 523

Hearst, Marti A., 452, 701

Hermjakob, Ulf, 389

Hirschberg, Julia, 807

Hovy, Eduard, 1048

Hoyt, Frederick, 326

Huang, Liang, 192, 586, 897

Hying, Christian, 496

Isozaki, Hideki, 665

Ji, Heng, 254

Jiampojarn, Sittichai, 905

Jiang, Hongfei, 559

Jiang, Wenbin, 897

Johnson, Mark, 398

Jung, Sangkeun, 630

Jurafsky, Dan, 380, 789

Kaisser, Michael, 701

Kaufmann, Tobias, 106

Kazama, Jun'ichi, 407

Kennedy, Alistair, 416

Kleeman, Alex, 959

Klein, Dan, 771, 879

Knight, Kevin, 389

Koehn, Philipp, 763

Koller, Alexander, 218

Koller, Daphne, 344

Kondrak, Grzegorz, 568, 905

Koo, Terry, 595

Kordoni, Valia, 613

Kozareva, Zornitsa, 1048

Lang, Jun, 843

Lapata, Mirella, 236, 272

Lee, Cheongjae, 630

Lee, Gary Geunbae, 630

Lee, John, 174

Lemon, Oliver, 638

Li, Chi-Ho, 89

Li, Guofu, 523

Li, Haizhou, 559

Li, Jianguo, 434

Li, Mu, 89

Li, Sheng, 559, 780, 843, 1021

Li, Zhifei, 425

Liang, Percy, 771, 879

Lin, Chin-Yew, 156, 710

Lin, Dekang, 10, 532, 994

Litman, Diane J., 622

Liu, Feifan, 541

Liu, Kang, 541

Liu, Qun, 192, 897

Liu, Ting, 780, 843, 1021

Louis, Annie, 825

Lowe, John B., 701

Lü, Yajuan, 897

Mairesse, François, 165

Mann, Gideon S., 870

Manning, Christopher D., 380, 959, 1039

Màrquez, Lluís, 550

Martinez, David, 317

Marton, Yuval, 1003

Matsuzaki, Takuya, 46

McCallum, Andrew, 870

McDonald, Ryan, 308, 950

Mehay, Dennis, 183

Mei, Qiaozhu, 816

Mi, Haitao, 192

Mihalcea, Rada, 932

Milidiú, Ruy Luiz, 647

Mitchell, Jeff, 236

Mitra, Pabitra, 488

Miyao, Yusuke, 46

Moore, Robert C., 97  
Moschitti, Alessandro, 798  
Mírovský, Jiří, 37  
Mrozinski, Joanna, 443  
Muresan, Smaranda, 1012

Nakov, Preslav, 452  
Nallapati, Ramesh, 245  
Namata, Galileo, 941  
Nenkova, Ani, 825  
Nesson, Rebecca, 604  
Ng, Hwee Tou, 55  
Ng, Raymond T., 353  
Nicholson, Jeremy, 613  
Nie, Jian-Yun, 148  
Niu, Cheng, 1021  
Nivre, Joakim, 950  
Nomoto, Tadashi, 299  
Nowak, Robert, 656

Oard, Douglas W., 461, 941  
Olsson, J. Scott, 461  
Osborne, Miles, 200

Paşca, Marius, 19, 994  
Penn, Gerald, 470  
Pfister, Beat, 106  
Pighin, Daniele, 798  
Polifroni, Joseph, 479

Quirk, Chris, 97

Rabbat, Michael, 656  
Rafferty, Anna N., 1039  
Rappoport, Ari, 227, 692, 861, 1030  
Ravichandran, Deepak, 674  
Regneri, Michaela, 218  
Reichart, Roi, 861, 1030  
Resnik, Philip, 1003, 1012  
Richman, Alexander E., 1  
Rieser, Verena, 638  
Riloff, Ellen, 1048  
Robertson, Stephen, 148  
Roth, Dan, 1030  
Roy, Deb, 121  
Ruopp, Achim, 514

Sagae, Kenji, 46

Saha, Sujan Kumar, 488  
Sammons, Mark, 1030  
Sarkar, Sudeshna, 488  
Satta, Giorgio, 604  
Scheible, Christian, 496  
Schmid, Helmut, 496  
Schone, Patrick, 1  
Schulte im Walde, Sabine, 496  
Schuurmans, Dale, 532  
Seneff, Stephanie, 174  
Sætre, Rune, 46  
Shen, Libin, 577  
Shieber, Stuart M., 604  
Snyder, Benjamin, 737  
Srikumar, Vivek, 1030  
Su, Jian, 843  
Surdeanu, Mihai, 719  
Suzuki, Hisami, 514  
Suzuki, Jun, 665  
Szarvas, György, 281  
Szpakowicz, Stan, 416  
Szpektor, Idan, 683

Talbot, David, 505  
Tan, Chew Lim, 559, 843  
Taskar, Ben, 986  
Thater, Stefan, 218  
Titov, Ivan, 308  
Tomanek, Katrin, 861  
Torisawa, Kentaro, 407  
Toutanova, Kristina, 514  
Tsarfaty, Reut, 371  
Tsuji, Jun'ichi, 46

Uszkoreit, Jakob, 755

Vadas, David, 335  
Van Durme, Benjamin, 19, 994  
Veale, Tony, 523  
Vickrey, David, 344  
Voorhees, Ellen M., 63  
Vozila, Paul, 114

Walker, Marilyn, 165, 479  
Wang, Haifeng, 780  
Wang, Qin Iris, 532  
Weerkamp, Wouter, 923

Weir, David, 968  
Weischedel, Ralph, 577  
White, Michael, 183  
Whittaker, Edward, 443

Xiong, Miao, 914  
Xu, Jia, 81  
Xu, Jinxi, 577

Yang, Fan, 541  
Yang, Xiaofeng, 843  
Yarowsky, David, 425  
Yu, Yong, 156, 914

Zapirain, Beñat, 550  
Zaragoza, Hugo, 719  
Zhai, ChengXiang, 816  
Zhang, Dongdong, 89  
Zhang, Hao, 97, 209  
Zhang, Min, 559  
Zhang, Yue, 888  
Zhao, Jun, 541  
Zhao, Shaojun, 994  
Zhao, Shiqi, 780, 1021  
Zhou, Ming, 89, 1021  
Zhou, Qi, 914  
Zhou, Xiaodong, 353  
Zhu, Xiaodan, 470  
Zhu, Xiaojin, 656  
Zhu, Xiaoyan, 710  
Zou, Bo, 541